Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9501 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9502 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9503 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9504 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9505 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9506 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9507 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9508 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9509 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9510 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9511 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9512 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9513 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9514 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9515 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9516 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9517 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9518 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9519 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9520 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?

Rodada de poker

Este exercício está baseado no problema 54 do excelente site `projecteuler.net`. No jogo de cartas **pôquer**, uma mão consiste em cinco cartas e elas são classificadas, da menor para a maior, da seguinte maneira:
Carta alta carta de maior valor.
Um par duas cartas do mesmo valor.
Dois pares dois pares diferentes.
Trinca três cartas do mesmo valor.
Straight todas as cartas têm valores consecutivos.
Flush todas as cartas do mesmo naipe.
Full House trinca e um par.
Quadra quatro cartas do mesmo valor.
Straight Flush todas as cartas são valores consecutivos do mesmo naipe.
Royal Flush Dez, Valete, Dama, Rei, Ás do mesmo naipe.
As cartas são avaliadas na seguinte ordem: 2, 3, 4, 5, 6, 7, 8, 9, 10, Valete, Dama, Rei, Ás.

Se dois jogadores tiverem as mesmas mãos classificadas, então a classificação composta pelo valor mais alto vence; por exemplo, um par de oitos vence um par de cincos (veja o exemplo 1 abaixo). Mas se duas classificações empatarem, por exemplo, ambos os jogadores tiverem um par de rainhas, então as cartas mais altas em cada mão são comparadas (veja o exemplo 4 abaixo); se as cartas mais altas empatarem, então as próximas cartas mais altas são comparadas, e assim por diante.

Neste jogo, os naipes recebem os nomes em inglês:
C=clubs=paus=♣
S=spaces=espadas=♠
H=hearts=copas=♡
D=diamonds=ouros= ♢
Os naipes são importantes nas análises do flush (5 cartas do mesmo naipe), straight flush (5 cartas do mesmo naipe consecutivas) e royal flush (um straight flush que começa em 10 e acaba em A).

Além disso, o naipe é irrelevante no poker. Ele não desempata nada, quem faz isso é o valor de face da carta em questão. Por exemplo, se um jogador tem uma trinca de K e outro uma trinca de 9, vence a trinca da carta mais alta (o rei). E, o que acontece se 2 jogadores fizerem straight flush (obviamente de naipes diferentes) ? Embora regulamentos locais possam determinar diferentemente, no geral a partida é empatada e as apostas divididas.

Considere as cinco mãos a seguir distribuídas a dois jogadores:

M	Jogador 1	Jogador 2	G
1	5H 5C 6S 7S KD Par de cincos	2C 3S 8S 8D TD par de oitos	2
2	5D 8C 9S JS CA Carta + alta As	2C 5C 7D 8S QH Carta + alta Rainha	1
3	2D 9C AS AH AC Três Ases	3D 6D 7D TD QD Flush de ouros	2
4	4D 6S 9H QH QC Par de Rainhas + Alta: Nove	3D 6D 7H QD QS Par de Rainhas + Alta: Sete	1
5	2H 2D 4C 4D 4S Full House com três quatros	3C 3D 3S 9S 9D Full House com três três	1

O código

```
global aa
aa=1
def compute():
  ans = sum(1 for handpair in HANDS if is_player1_win(handpair))
  return str(ans)
def is_player1_win(handpair):
  global aa
  cards = [parse_card(item) for item in handpair.split(" ")]
  assert len(cards) == 10
  player1 = cards[ : 5]
  player2 = cards[5 : ]
  aa=aa+1
  return get_score(player1) > get_score(player2)
def get_score(hand):
  assert len(hand) == 5
    rankcounts = [sum(1 for (rank, _) in hand if rank == i) for i in range(13)]
    rankcounthist = [rankcounts.count(i) for i in range(6)]
    minsuit = min(suit for (_, suit) in hand)
    maxsuit = max(suit for (_, suit) in hand)
    flushsuit = minsuit if minsuit == maxsuit else -1
    bestcards = get_5_frequent_highest_cards(rankcounts, rankcounthist)
    straighthighrank = get_straight_high_rank(rankcounts)
      if   straighthighrank != -1 and flushsuit != -1     : return 8 << 20 | straighthighrank  # Straight flush
      elif rankcounthist[4] == 1                          : return 7 << 20 | bestcards         # Four of a kind
      elif rankcounthist[3] == 1 and rankcounthist[2] == 1: return 6 << 20 | bestcards         # Full house
      elif flushsuit != -1                                : return 5 << 20 | bestcards         # Flush
      elif straighthighrank != -1                         : return 4 << 20 | straighthighrank  # Straight
      elif rankcounthist[3] == 1                          : return 3 << 20 | bestcards         # Three of a kind
      elif rankcounthist[2] == 2                          : return 2 << 20 | bestcards         # Two pairs
      elif rankcounthist[2] == 1                          : return 1 << 20 | bestcards         # One pair
      else                                                : return 0 << 20 | bestcards         # High card
def get_5_frequent_highest_cards(ranks, rankshist):
  result = 0
  count = 0
  for i in reversed(range(len(rankshist))):
    for j in reversed(range(len(ranks))):
      if ranks[j] == i:
        for k in range(i):
          if count >= 5:
            break
          result = result << 4 | j
          count += 1
  if count != 5:
    raise ValueError()
  return result

def get_straight_high_rank(ranks):
  for i in reversed(range(3, len(ranks))):
    for j in range(5):
      if ranks[(i - j + 13) % 13] == 0:
        break
    else:
      return i
  return -1

def parse_card(card):
  return (RANKS.index(card[0]), SUITS.index(card[1]))


RANKS = "23456789TJQKA"
SUITS = "SHCD"


HANDS = [
 "8C TS KC 9H 4S 7D 2S 5D 3S AC",
 "5C AD 5D AC 9C 7C 5H 8D TD KS",
 "3H 7H 6S KC JS QH TD JC 2D 8S",
 "TH 8H 5...

... aqui entram as cartas de seu arquivo...

print(compute())  # AQUI ELE IMPRIME A QTD DE VITORIAS DO 1
```

☞ Para você fazer

O arquivo FH9521 publicado no local de sempre, contém 200 mãos aleatórias distribuídas para dois jogadores. Cada linha do arquivo contém dez cartas (separadas por um único espaço): as cinco primeiras são cartas do Jogador 1 e as cinco últimas são cartas do Jogador 2. Você pode assumir que todas as mãos são válidas (sem caracteres inválidos ou cartas repetidas), a mão de cada jogador não está em nenhuma ordem específica.

Quantas mãos o Jogador 1 ganha?