=

Pedro Kantek

V MARATONA
UP DE PROGRAMAÇÃO
(2015)

Preliminares

A maratona vai acontecer (salvo melhor juízo) dia 12/11/2015, nas dependências da UP no campus do Campo Comprido. Para efeito de treinamento, deve-se incluir os problemas da maratona de 2013. Para 2015, haverá apenas uma categoria e os problemas a incluir são:

1: MULNAT - Múltiplo dos Naturais

Os números naturais são aqueles que usamos nas contagens de um modo geral. Diz-se que a é múltiplo de b se quando se divide b por a obtem-se um resto igual a zero. Assim, por exemplo, 21 é múltiplo de 7 pois ao fazer $21 \div 7 = 3$ e não ocorre resto nessa divisão.

Se listarmos os números naturais abaixo de 20 que são múltiplos de 3 ou 5 ou 7 ou 11 obteremos 3 5 6 7 9 10 11 12 14 15 18 cuja soma é 110.

Note que o fato do mesmo número ser múltiplo de mais de um valor (como por exemplo, o 15 que é múltiplo de 3 e também de 5) não implica que ele seja somado várias vezes.

Assim, um determinado número ou é múltiplo de um ou mais valores (quando então ele é somado) ou não é múltiplo de nenhum dos valores (quando então ele não é somado).

Pede-se que seja escrito um programa de computador que, quando executado, receba na entrada um único número indicando o limite abaixo do qual os múltiplos de 3, 5, 7 ou 11, devem ser somados.

Entrada

Uma série de valores inteiros. Cada um sempre é maior que 1 e menor que 1000000 e indica o limite abaixo do qual a soma de múltiplos de 3,5,7 ou 11 deve ser calculada. O último número da série é 0 e este número sinaliza o final dos dados e não deve ser processado.

Saída

A soma dos múltiplos de 3,5,7 ou 11, abaixo da entrada. O número deve ser formatado como inteiro sem nenhum tipo de separação entre milhares.

Exemplo

Para as seguintes entradas

```
1000
2000
999999
0
```

Devem ser obtidas as seguintes saídas

```
292285
1168230
292206707821
```

DADOS ESCONDIDOS

Dados para montagem dos arquivos de validação no ambiente BOCA

```
123456 -> 4453637505
990000 -> 286391937853
10 -> 30
2 -> 0
555555 -> 90187132035
0 -> fim dos dados de entrada
```

Solução em C

```
#include<stdio.h>
int main() {
 long long int lim,som,i;
 scanf("%lld",&lim);
 while (lim != 0) {
  som = 0;
  for (i=1;i<lim;i++) {
   if (((i%3)==0)||((i%5)==0)||((i%7)==0)||((i%11)==0)) {
     som = som + i;
}
 }
 printf("%lld\n",som);
 scanf("%lld",&lim);
 }
return 0;
}
```

Solução em APL2

```
  [0] mar15a lim;x
  [1] ⍝ ((0=3|x)∨(0=5|x)∨(0=7|x)∨(0=11|x))/x←1+⍳lim-2
  [2] 20 0⍕+/((0=3|x)∨(0=5|x)∨(0=7|x)∨(0=11|x))/x←1+⍳lim-2
```

2: COEFBI - Coeficientes Bionomiais

O símbolo $\begin{pmatrix} n \\ r \end{pmatrix}$ que é lido como n C r onde r e n são inteiros positivos com $r \leq n$ é definido como segue

$$\begin{pmatrix} n \\ r \end{pmatrix} = \frac{n \times (n-1) \times (n-2) \times ... \times (n-r+1)}{1 \times 2 \times 3... \times (r-1) \times r}$$

Estes números são denominados coeficientes binomiais. Alguns exemplos
$\begin{pmatrix} 8 \\ 2 \end{pmatrix}$=$\frac{8\times7}{1\times2}$ =28; $\begin{pmatrix} 9 \\ 4 \end{pmatrix}$=$\frac{9\times8\times7\times6}{1\times2\times3\times4}$ =126;
$\begin{pmatrix} 12 \\ 5 \end{pmatrix}$=$\frac{12\times11\times10\times9\times8}{1\times2\times3\times4\times5}$ =792; $\begin{pmatrix} 13 \\ 1 \end{pmatrix}$=$\frac{13}{1}$ =13

Deve-se notar que o coeficiente binomial encontra múltiplas aplicações em vários ramos da matemática. Com ele faz-se uma construção engenhosa chamada Triângulo de Pascal, que lista os coeficientes na expansão da exponenciação de binômios.

Espera-se que você escreva um programa que leia pares de números r e n e para cada par lido, calcule e imprima o coeficiente binomial $\begin{pmatrix} n \\ r \end{pmatrix}$.

Deve-se notar que para este cálculo há uma fração, na qual há r termos tanto no numerador quanto no denominador da fração. Igualmente, deve-se perceber que a divisão sempre resulta um número inteiro.

Entrada

Pares de números inteiros n e r, com $n \geq r$, separados por um único caractere em branco e ambos menores do que 100. A quantidade de pares a ler é indeterminada e será encerrada quando for lido um par que tenha $n < r$. Este último par não deve ser processado e ele sinaliza fim de dados.

Saída

Para cada par lido, o programa deve calcular e imprimir em uma nova linha o coeficiente binomial pedido. O número deve ser impresso sem nenhuma formatação especial.

Exemplos

```
25 6
27 13
12 8
34 17
33 22
34 14
9 4
31 14
15 4
3 4
```

Para essas entradas as saídas devem ser

```
177100
20058300
495
2333606220
193536720
1391975640
126
265182525
1365
```

DADOS ESCONDIDOS

```
40 18 ->      113380261800
28 19 ->           6906900
22 4 ->               7315
26 10 ->           5311735
42 19 ->      446775310800
19 11 ->             75582
```

Solução em C

```
#include<stdio.h>
int main() {
long long int r,n,res;
scanf("%lld %lld",&n,&r);
while (n>r) {
   long long q,num,den;
   num=1;
   den=1;
   res=1;
   for (q=1;q<=r;q++){
     num=(num*n)/q;
 n--;
 }
    printf("%lld\n",num);
    scanf("%lld %lld",&n,&r);
}
return 0;
}
```

Solução em APL2

```
a!b
```

3: MASFEM - Masculino e Feminimo

As linguagens mais comuns no mundo dos negócios (como C e seus dialetos, Java, Pascal, Clipper, etc) tratam frases de texto como sendo um vetor de caracteres. Esta não é a única maneira de processar uma frase. Por exemplo, LISP trata uma frase como sendo uma lista e nele cada palavra é um elemento atômico, indivisível e portanto muito mais fácil de ser manuseado.

Suponha que foi criada uma linguagem artificial na qual as palavras terminadas pela letra O são consideradas masculinas. Nessa linguagem, o feminino de uma palavra masculina sempre é obtido trocando-se o O final pela letra A, sem nenhuma outra alteração.

Pede-se que você construa um programa que receba uma frase escrita nessa linguagem e que reescreva a frase, transformando as palavras masculinas em sua palavra feminina equivalente.

Entrada

Um conjunto de frases, todas escritas em letra maiúscula. Cada frase é encerrada por um ponto. As palavras são separadas entre si por uma única posição em branco. O comprimento máximo de cada frase, o ponto incluído, é de 80 caracteres. A última frase é `X.` que não deve ser processada e que determina o final do processamento.

Saída

A frase lida deve ser processada e o resultado impresso.

Exemplos

Supondo que o programa seja chamado com as seguintes frases

```
OUVIRAM DO IPIRANGA AS MARGENS PLACIDAS DE UM POVO O HEROICO.
IVO VIU A UVA.
IVO VIU A UVA E O POLO.
CURITIBA PARANA BRASIL.
ONCOTO PONCOVO.
OI.
X.
```

Deve gerar a saída seguinte

```
OUVIRAM DA IPIRANGA AS MARGENS PLACIDAS DE UM POVA A HEROICA.
IVA VIU A UVA.
IVA VIU A UVA E A POLA.
CURITIBA PARANA BRASIL.
ONCOTA PONCOVA.
OI.
```

DADOS ESCONDIDOS

```
ASDRUBAL TROUXE O TROMBONE. -> ASDRUBAL TROUXE A TROMBONE.
O TOCO DO CARRO CAIU. -> A TOCA DA CARRA CAIU.
O OO OOO OOOO OOOOO. -> A OA OOA OOOA OOOOA.
SAO TODOS FARINHA DO MESMO SACO E DA MESMA SACOLA. ->
  SAA TODOS FARINHA DA MESMA SACA E DA MESMA SACOLA.
```

Solução em C

```
#include<stdio.h>
int main() {
unsigned char fr[80],frs[80];
gets(fr);
while ((fr[0]!='X')||(fr[1]!='.')) {
   int i = 0;
   while (fr[i]!='.') {
      frs[i]=fr[i];
      i++;
   }
   frs[i]='.';
   i++;
   frs[i]='\0';
   int j = 0;
   while (j<i) {
      if (((fr[j]=='O')&&(fr[j+1]==' '))||((fr[j]=='O')&&
                    (fr[j+1]=='.'))) {
     frs[j]='A';
  }
  j++;
   }
   puts(frs);
   gets(fr);
}
return 0;
}
```

Solução em APL

```
  [0] y←frasart x;i
  [1] y←x
  [2] i←2
  [3] t1:→(x[i]='.')/fim
  [4] →((x[i]=' ')^(x[i-1]='O'))/troca
  [5] y[i-1]←x[i-1]
  [6] →soma
  [7] troca:
  [8] y[i-1]←'A'
  [9] soma:
 [10] i←i+1
 [11] →t1
```

```
[12] fim:→(x[i-1]≠'O')/0
[13] y[i-1]←'A'
```

4: GENOMA - Projeto Genoma

Desde os trabalhos pioneiros de Darwin (começo do século XIX) e Mendel (meados do mesmo século) avançou-se bastante na compreensão do processo pelo qual a vida surgiu e se modificou ao longo dos últimos 3 bilhões de anos em nosso planeta.

Mais recentemente, nos anos 50 do século passado, dois pesquisadores (Watson e Crick) descreveram a molécula do DNA (Ácido Desoxirribonucleico) como o elemento que carrega a herança entre as gerações de seres vivos e que em última análise faz com que um filhote de elefante seja um elefantinho e não uma minhoca.

Para mapear todo o material genético do ser humano desenvolve-se o Projeto Genoma Humano. As moléculas de DNA (moléculas que contêm material genético) podem ser representadas por cadeias de caracteres que usam um alfabeto de apenas 4 letras: 'A', 'C', 'T' e 'G'. Um exemplo de uma tal cadeia é:

```
TCATATGCAAATAGCTGCATACCGA
```

Nesta tarefa você deverá produzir uma ferramenta muito utilizada no projeto Genoma: um programa que procura ocorrências de uma pequena cadeia de DNA (que vamos chamar de p) dentro de uma outra cadeia de DNA (que vamos chamar de t). Você deverá procurar dois tipos de ocorrência: a "direta" e a "complementar invertida". Uma ocorrência *direta* é quando a cadeia p aparece como subcadeia dentro de t. Por exemplo, se

```
p = CATA          e          t = TCATATGCAAATAGCTGCATACCGA
```

então p ocorre na forma direta na posição 2 e na posição 18 de t. Uma ocorrência *complementar invertida* depende da seguinte correspondência entre as letras do DNA: 'A' $\iff$ 'T' e 'G' $\iff$ 'C'. "Complementar o DNA" significa trocar as letras de uma cadeia de DNA seguindo essa correspondência. Se complementarmos a cadeia CATA, vamos obter GTAT. Mas além de complementar, é preciso também inverter, ou seja, de GTAT obter TATG. E é esta cadeia que deverá ser procurada, no caso da ocorrência complementar invertida. Assim, se p e t são as mesmas cadeias do exemplo anterior, então p ocorre na forma complementar invertida na posição 4 de t.
Sua tarefa é escrever um programa que, dadas duas cadeias p e t, onde o comprimento de p é menor ou igual ao comprimento de t, procura todas as ocorrências diretas e todas as ocorrências complementares invertidas de p em t.

Entrada de Dados: A entrada contém vários conjuntos de teste. Cada conjunto de teste é composto por três linhas. A primeira linha contém dois inteiros positivos, M e N, M N, que indicam respectivamente o comprimento das cadeias de DNA p e t, conforme descrito acima. A segunda linha do conjunto de teste contém a cadeia p, e a terceira linha contém a cadeia t, onde p e t são compostas utilizando apenas os caracteres 'A', 'C', 'G' e 'T'. O final do arquivo de testes é indicado quando M = N = 0 (este último conjunto de testes não é válido e não deve ser processado). Tanto p quanto t serão menores do que 100 caracteres.

```
Exemplo de Entrada
2 4
AC
TGGT
4 25
CATA
TCATATGCAAATAGCTGCATACCGA
0 0
```

Saída de Dados: Seu programa deve produzir uma saída onde para cada conjunto de teste do arquivo de entrada seu programa deve produzir duas linhas. Na primeira linha deve aparecer a lista, em ordem crescente, com a posição inicial de cada ocorrência, na forma direta, do padrão p na seqüência t. Na segunda linha deve aparecer a lista, em ordem crescente, com a posição inicial de cada ocorrência, na forma complementar invertida, do padrão p na seqüência t. Note a ausência de pontuação.
Exemplo de Saída:

```
Teste 1
ocorrencia direta: 0
ocorrencia complementar invertida: 3
Teste 2
ocorrencia direta: 2 18
ocorrencia complementar invertida: 4
```

(esta saída corresponde ao exemplo de entrada acima) Este exercício é igual ao proposto na OBI99.

Exemplos

```
   4 35
 TGCT
 GCTAAGGTTATTAAGTGCTGCTAAGCAGGCATGTC
   5 50
 ATGTG
 TCAGGTTAATGTGTAAACCGAATCACATTCCCGCTTCATGTGGTACGCAT
   4 37
 CTGG
 ATGGTCTGGCTTACGGCCAGCCGGTTGCTGGATGAGA
   5 44
 TTACG
 TAGTAGTTCGGCGTACGTAAGCGTTACGATGACGCGGACTTGCG
0 0
```

para esta entrada a saída deverá ser (atente para as frases e a formatação)

```
Teste 1
ocorrencia direta: 16 19
ocorrencia complementar invertida: 24
Teste 2
ocorrencia direta: 9 38
ocorrencia complementar invertida: 24
Teste 3
ocorrencia direta: 6 28
ocorrencia complementar invertida: 17
Teste 4
ocorrencia direta: 24
ocorrencia complementar invertida: 16
```

DADOS ESCONDIDOS

```
   5 39
 CCATC
 AAGAGAAAAGATGGTCGCCATCTTGTTCGAACCCGTCAA
   7 43
 TTCCGCG
 TAATAATGCAGGTCGCCACAACTTCCGCGTTCGCGGAATTTCG
   5 36
 GTCTG
 GACAAGGTTGTCTGTCAGACGTGCAGGACCGAGACC
   5 44
 GCCTC
 GTTAAAAGCCTCCATACATTTCGTCTGAGGCGCCTCCGCTTCCA
0 0
                      ->
Teste 1
ocorrencia direta: 18
ocorrencia complementar invertida: 10
Teste 2
ocorrencia direta: 23
ocorrencia complementar invertida: 32
Teste 3
ocorrencia direta: 10
ocorrencia complementar invertida: 16
Teste 4
ocorrencia direta: 8 32
ocorrencia complementar invertida: 27
```

Em C

```
#include<stdio.h>
int main() {
int tamp,tamt;
int i,achou,j;
int vd[100],vi[100];
int qtd,qti,nvez;
unsigned char p[80],t[80],pi[80],aux[80];
scanf("%d %d \n",&tamp,&tamt);
nvez=1;
while ((tamp!=0)||(tamt!=0)) {
   qtd=0;
   qti=0;
   gets(p);
   gets(t);
   for (i=0;i<tamp;i++){
      if(p[i]=='A') { aux[i]='T';}
      if(p[i]=='T') { aux[i]='A';}
      if(p[i]=='C') { aux[i]='G';}
      if(p[i]=='G') { aux[i]='C';}
   }
```

```
   for (i=1;i<=tamp;i++){
      pi[i-1]=aux[tamp-i];
   }
   for(i=0;i<tamt;i++) {
      if (t[i]==p[0]){
     achou=1;
 for(j=1;j<tamp;j++) {
    if(t[i+j]!=p[j]){
   achou=0;
}
         }
         if (achou==1) {
            vd[qtd]=i+1;
qtd++;
 }
       }
   achou=0;
   if (t[i]==pi[0]){
     achou=1;
 for(j=1;j<tamp;j++) {
    if(t[i+j]!=pi[j]){
   achou=0;
}
         }
         if (achou==1) {
           vi[qti]=i+1;
       qti++;
         }
       }
   }
   printf("Teste %d\n",nvez);
   nvez++;
   printf("Ocorrencia direta:");
   if (qtd==0) {
      printf(" 0\n");
  }
   else {
     for(i=0;i<qtd;i++){
        printf(" %d",vd[i]);
 }
 printf("\n");
   }
   printf("Ocorrencia complementar invertida:");
   if (qti==0) {
      printf(" 0\n");
  }
   else {
     for(i=0;i<qti;i++){
        printf(" %d",vi[i]);
 }
```

```
 printf("\n");
   }
scanf("%d %d \n",&tamp,&tamt);
}
}
```

Em APL2

```
 r←p resolvegenoma t;x;y;plinha;z;w
⍝ busca ocorrencias diretas e inversas de p em t
 r←0 60⍴''
 x←(p⍷t)/⍳⍴t
 r←r,[1]1 60⍴60↑'Ocorrencias diretas=',⍕x
 plinha←'Ax' edit p
 plinha←'Gy' edit plinha
 plinha←'TA' edit plinha
 plinha←'CG' edit plinha
 plinha←'xT' edit plinha
 plinha←'yC' edit plinha
 plinha←⍉plinha
 y←(plinha⍷t)/⍳⍴t
 r←r,[1]1 60⍴60↑'Ocorr. complementar invertida=',⍕y
 z←((⍉plinha)⍷t)/⍳⍴t
 r←r,[1]1 60⍴60↑'ERRO: complementar=',⍕z
 w←((⍉p)⍷t)/⍳⍴t
 r←r,[1]1 60⍴60↑'ERRO: invertida=',⍕w
```

5: FIBONA - Fibonacci

No século XIII viveu na Itália, o matemático Leonardo Bigollo, também conhecido como Leonardo Fibonacci, por muitos considerado o maior matemático da idade média na Europa. Em 1202, com a idade de 32 anos, ele escreveu o livro *Liber Abaci* (O livro do ábaco ou o Livro de cálculo) no qual ele descreveu e usou os numerais indo-arábicos e começou assim o processo de introdução desses numerais na nossa cultura, onde até hoje são usados. Nesse livro, ele também descreveu e usou a assim chamada Sequência de Fibonacci, cuja definição é

$$\begin{cases} F(1) & = 1 \\ F(2) & = 1 \\ F(n) & = F(n-1) + F(n-2) \quad se\ n \geq 3 \end{cases}$$

é uma sequência de números que cresce muito rápido. Ela pode ser obtida facilmente escrevendo um programa de computador recursivo que implemente facilmente a definição acima. Por exemplo, fazendo (função fibo, em C)

```
int function fibo(int X) {
  if X < 3
    return 1
  else
    return (fibo X-1) + fibo (X-2)
  endif
}
```

O programa acima, funciona, mas tem um problema: o recálculo sucessivo de instância que já foram calculadas. Apenas para ilustrar, usando um computador bem rapidinho, para calcular fibo(50) o programa acima demora mais de 1 dia.

Então, você deve construir um programa para calcular a sequência de Fibonacci, mas deve fugir da implementação acima.

Seu programa receberá um número n e deverá imprimir o $n - esimo$ número de Fibonacci.

Entrada

Uma série de números inteiros, um em cada linha, todos eles menores do que 47. O último número, e que não deve ser processado será maior ou igual do que 47 e ele indica final dos dados.

Saída

A cada número lido, o programa deve imprimir o número de Fibonacci equivalente, sem nenhum tipo especial de formatação.

Exemplos

Para a entrada

```
1
11
21
31
41
1001
```

Deverá haver a seguinte saída

```
                1
               89
            10946
          1346269
        165580141
```

DADOS ESCONDIDOS

```
45 ->    1134903170
33 ->       3524578
32 ->       2178309
2 ->              1
3 ->              2
4 ->              3
5 ->              5
10 ->            55
15 ->           610
20 ->          6765
46 ->    1836311903
47
```

Em C

```
#include<stdio.h>
long int fibo[47];
int main() {
int n,i;
for (i=1;i<=46;i++) {
   fibo[i]=0;
   }
fibo[1]=1;
fibo[2]=1;
scanf("%d",&n);
while (n<47) {
   printf("%ld\n",achafib(n));
   scanf("%d",&n);
}
return (0);
}
long int achafib(int nn) {
   if (fibo[nn] == 0) {
  fibo[nn] = achafib(nn-1) + achafib(nn-2);
   }
   return (fibo[nn]);
}
```

Em APL2

```
  [0] criam
  [1] mfib←1 1,998⍴-1

  [0] r←fibrecu n
  [1] →(n<3)/um
  [2] r←(fibrecu(n-1))+fibrecu(n-2)
  [3]
  [4] →0
  [5] um:r←1
```

```
[0] r←fibmemoizing n
[1] →(mfib[n]≠-1)/achou
[2] mfib[n]←(fibmemoizing(n-1))+fibmemoizing(n-2)
[3] achou:r←mfib[n]
```

6: MELALU - Melhor Aluno

Em uma determinada disciplina de um curso de informática a cada aula os alunos recebem um exercício individual que deve ser respondido e devolvido ao professor em até 2 semanas de prazo. No final do bimestre, a média de todos os exercícios entra no cálculo do bimestre com peso 60 enquanto a prova entra com o peso 40.

Supondo turmas de 5 alunos e bimestres com 6 aulas de exercícios, você vai receber um conjunto de 5 linhas e 7 colunas contendo as notas do bimestre para aquela turma. A primeira coluna dos dados representa a nota da prova e os outros 6 valores da linha representam as notas dos 6 exercícios daquele aluno.

Seu programa deve calcular as 5 notas e imprimir 'A', se o primeiro aluno tiver a melhor média, 'B' se a melhor média for do segundo aluno, 'C' para o terceiro, 'D' para o quarto e 'E' para o quinto.

Entrada

O primeiro número isolado na primeira linha é um inteiro que informa quantas turmas vem a seguir. Cada turma tem 5 alunos e portanto será formada por 5 linhas de 7 números reais (menores ou iguais a 10.0) com eventualmente uma decimal. Os dados da primeira linha correspondem ao aluno 'A'. Os da segunda a 'B' e assim por diante: os dados da última linha correspondem ao aluno 'E'. Não vai ocorrer nenhum caso de empate nos dados de entrada.

Saída

Um único caractere 'A'..'E' indicando qual aluno conseguiu a melhor média naquela turma.

Exemplos

Seja a seguinte entrada

```
4
0   6   2   8    1.5 3   0.5
9.5 4.5 2   4.5  2.5 8.5 2
1   9   1   6.5 10   6   1
1.5 5   8.5 7    5   1   5.5
6.5 1   9.5 9    0   6   7
8.5 9   6.5 6.5 0   9   2
9   8.5 6   0   8.5 2   0
5   9.5 1   2.5 5.5 2   1
0   9   9   2   7   5   2.5
3   6.5 2   2   3   0.5 3.5
 1.5 4   10   2   1   4   2
 7.5 0    2.5 1.5 9   5.5 8.5
10   5    7   6.5 7   0   2.5
 0.5 2.5  9   4.5 5   6   5
 1   3.5  4.5 7   4.5 3   6.5
1.5 5   4.5 10   10    4    8.5
4   7.5 4.5  4    6.5  5.5  0.5
0   1.5 3    7.5  3    5   10
4.5 4.5 6    6    4.5 10    2
7   5.5 2    3    8    9.5  2
```

Que produziu essa saída

```
B
A
C
E
```

DADOS ESCONDIDOS

```
5
9   2   1.5 8   9.5 8.5 5.5
6   1.5 8.5 9.5 6   7   2
8   4.5 0   6.5 6.5 1.5 1
2.5 8   5.5 4.5 5.5 1.5 6.5
0.5 5   3.5 7   8.5 4   2.5
0.5 0.5 1   10   4 2.5 0
9   6.5 5.5  3.5 2 1.5 9.5
1.5 0   7.5  5   4 0.5 0
3.5 4   8.5  9.5 3 5.5 3.5
5.5 5   0    5.5 8 6   9.5
0    2    6    2 5.5 1.5 4
5.5 10   10    6 6.5 0   5
8.5  4.5  6.5 10 7.5 9.5 3.5
5.5  7.5  3    4 5   4.5 3
1.5  6.5  8    7 0   2.5 5.5
 0   5.5 3   4    8   5   5.5
 3   5.5 9   3.5  3.5 1.5 9
 0   9   4.5 4.5  4.5 3.5 7
10   1   2.5 0    6   9   6.5
 9.5 0   3   7.5 10   4.5 0.5
5.5  3 0.5  5.5 2   4   9.5
1.5 10 6.5  3.5 8   2   5.5
1    5 1    3.5 6   7.5 3.5
0.5  8 9.5 10   7.5 2   8.5
7    1 4.5  9.5 5.5 9.5 7.5
-> ABCDE
```

Em C

```
#include<stdio.h>
int main() {
int n,i;
float a1,a2,a3,a4,a5,a6,a7;
float b1,b2,b3,b4,b5,b6,b7;
float c1,c2,c3,c4,c5,c6,c7;
float d1,d2,d3,d4,d5,d6,d7;
float e1,e2,e3,e4,e5,e6,e7;
float ma,mb,mc,md,me;
scanf("%d",&n);
for (i=1;i<=n;i++) {
 scanf("%f %f %f %f %f %f %f",&a1,&a2,&a3,&a4,&a5,&a6,&a7);
 scanf("%f %f %f %f %f %f %f",&b1,&b2,&b3,&b4,&b5,&b6,&b7);
 scanf("%f %f %f %f %f %f %f",&c1,&c2,&c3,&c4,&c5,&c6,&c7);
 scanf("%f %f %f %f %f %f %f",&d1,&d2,&d3,&d4,&d5,&d6,&d7);
 scanf("%f %f %f %f %f %f %f",&e1,&e2,&e3,&e4,&e5,&e6,&e7);
 ma = (a1*4)+a2+a3+a4+a5+a6+a7;
 mb = (b1*4)+b2+b3+b4+b5+b6+b7;
 mc = (c1*4)+c2+c3+c4+c5+c6+c7;
```

```
 md = (d1*4)+d2+d3+d4+d5+d6+d7;
 me = (e1*4)+e2+e3+e4+e5+e6+e7;
 if ((ma > mb) && (ma > mc) && (ma > md) && (ma > me)) {
    printf("A\n");}
 if ((mb > ma) && (mb > mc) && (mb > md) && (mb > me)) {
    printf("B\n");}
 if ((mc > ma) && (mc > mb) && (mc > md) && (mc > me)) {
    printf("C\n");}
 if ((md > ma) && (md > mb) && (md > mc) && (md > me)) {
    printf("D\n");}
 if ((me > ma) && (me > mb) && (me > mc) && (me > md)) {
    printf("E\n");}
 }
return 0;
}
```

Em APL2

```
  [0] r←faz4 q;x;y;in
  [1] novo:
  [2] x←5 7⍴¯0.5+(?35⍴21)÷2
  [3] y←(x[;1]×4)++/x[;1+⍳6]
  [4] ⍝ x←x,5 1⍴y
  [5] y←y[in←⍋y]
  [6] →((y[1]=y[2])∨(y[2]=y[3])∨(y[3]=y[4])∨(y[4]=y[5]))/novo
  [7] x←x[in;]
  [8] →(q='A')/ea
  [9] →(q='B')/eb
 [10] →(q='C')/ec
 [11] →(q='D')/ed
 [12] →(q='E')/ee
 [13] →0
 [14] ea:r←x[5 4 3 2 1;]
 [15] →0
 [16] eb:r←x[1 5 2 3 4;]
 [17] →0
 [18] ec:r←x[1 4 5 3 2;]
 [19] →0
 [20] ed:r←x[1 3 2 5 4;]
 [21] →0
 [22] ee:r←x[3 2 1 4 5;]
 [23] →0
```