Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

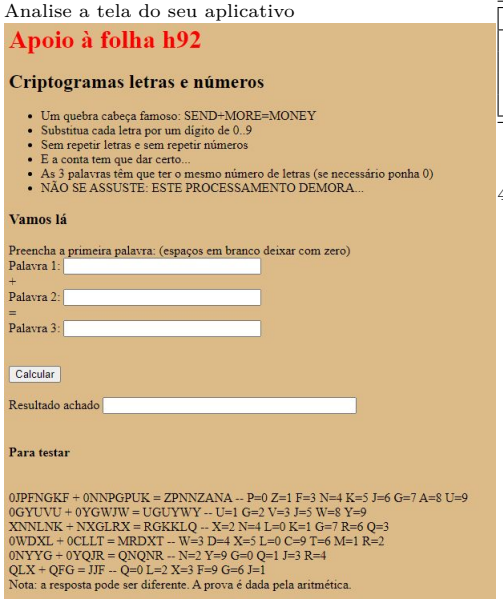
abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

Analise a tela do seu aplicativo



Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

```
  UVZFEM
+ VFDVUS
--------
 MJDEVDM
```

Responda aqui:

valores das letras

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''')))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

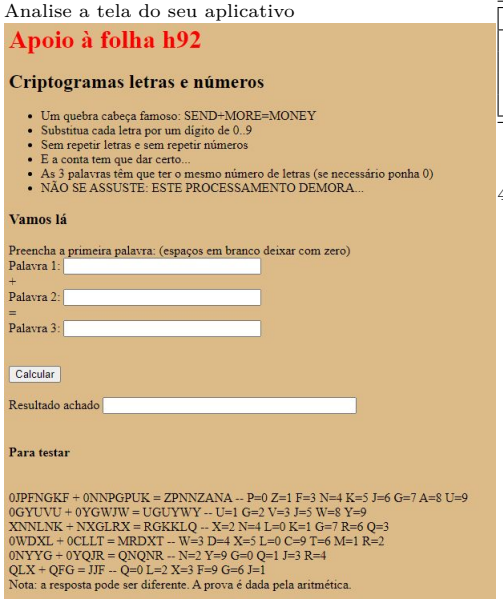
Depois que programar o aplicativo calcule a solução para o criptograma

```
  ZEGSDJ
+ UEDQET
--------
  QTDJEQ
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

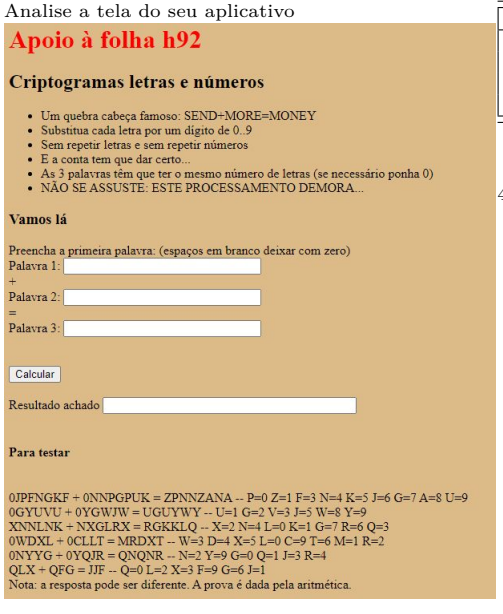
```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Analise a tela do seu aplicativo



Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

```
  ELXBXN
+ NECAER
--------
 RRVACBB
```

Responda aqui:

valores das letras

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 16/09/2024 - 09:49:42.0
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
Criptograma letras e números VIVXh92a V: 1.02
76025 ERIC CORDEIRO
24FRO408 - 4 _____ / _____ / _____

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

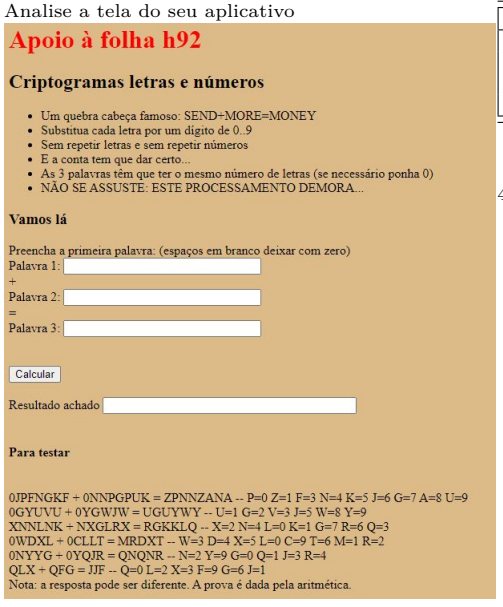
```
  BUBZBR
+ YBRRTU
--------
  LRBBZU
```

Responda aqui:

valores das letras

408-76025 - /

Analise a tela do seu aplicativo



Apoio à folha h92

Criptogramas letras e números

- Um quebra cabeça famoso: SEND+MORE=MONEY
- Substitua cada letra por um dígito de 0..9
- Sem repetir letras e sem repetir números
- E a conta tem que dar certo...
- As 3 palavras têm que ter o mesmo número de letras (se necessário ponha 0)
- NÃO SE ASSUSTE: ESTE PROCESSAMENTO DEMORA...

Vamos lá

Preencha a primeira palavra: (espaços em branco deixar com zero)
Palavra 1:
+
Palavra 2:
=
Palavra 3:

Calcular

Resultado achado

Para testar

0JPFNGKF + 0NNPGPUK = ZPNNZANA -- P=0 Z=1 F=3 N=4 K=5 J=6 G=7 A=8 U=9
0GYUVU + 0YGWJW = UGUYWY -- U=1 G=2 V=3 J=5 W=8 Y=9
XNNLNK + NXGLRX = RGKKLQ -- X=2 N=4 L=0 K=1 G=7 R=6 Q=3
0WDXL + 0CLLT = MRDXT -- W=3 D=4 X=5 L=0 C=9 T=6 M=1 R=2
0NYYG + 0YQJR = QNQNR -- N=2 Y=9 G=0 Q=1 J=3 R=4
QLX + QFG = JJF -- Q=0 L=2 X=3 F=9 G=6 J=1
Nota: a resposta pode ser diferente. A prova é dada pela aritmética.

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''')))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

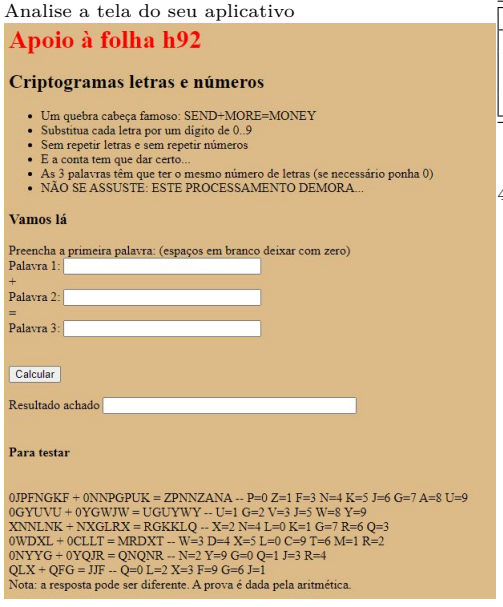
Depois que programar o aplicativo calcule a solução para o criptograma

```
  EWTWQW
+ KZJUJJ
--------
  TQJZGP
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 16/09/2024 - 09:49:42.0
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
Criptograma letras e números VIVXh92a V: 1.02
76049 FELIPE JESMIEL LEITE
24FRO408 - 6 _____ / _____ / _____

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

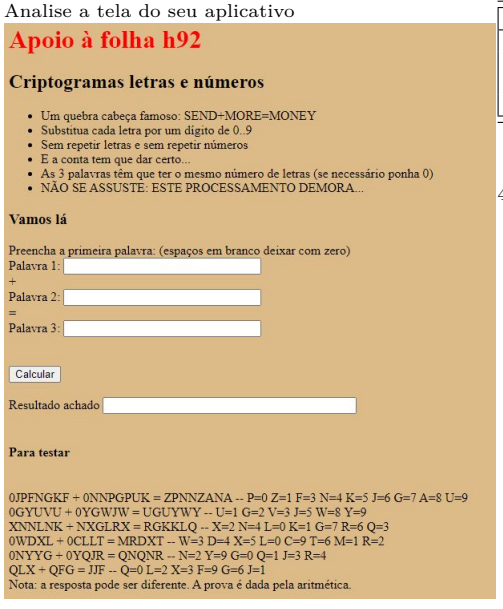
```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
```

Analise a tela do seu aplicativo




```
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

```
  RYCERF
+ ENCFNC
--------
 DBNRRED
```

Responda aqui:

valores das letras

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

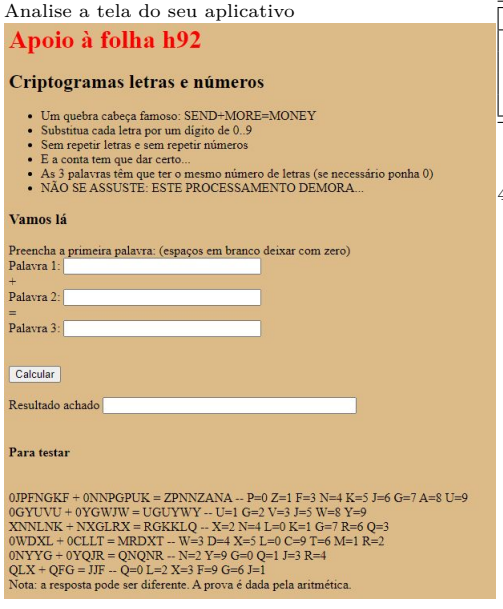
```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Analise a tela do seu aplicativo



Apoio à folha h92

Criptogramas letras e números

- Um quebra cabeça famoso: SEND+MORE=MONEY
- Substitua cada letra por um dígito de 0..9
- Sem repetir letras e sem repetir números
- E a conta tem que dar certo...
- As 3 palavras têm que ter o mesmo número de letras (se necessário ponha 0)
- NÃO SE ASSUSTE: ESTE PROCESSAMENTO DEMORA...

Vamos lá

Preencha a primeira palavra: (espaços em branco deixar com zero)
Palavra 1:
+
Palavra 2:
=
Palavra 3:

Calcular

Resultado achado

Para testar

0JPFNGKF + 0NNPGPUK = ZPNNZANA -- P=0 Z=1 F=3 N=4 K=5 J=6 G=7 A=8 U=9
0GYUVU + 0YGWJW = UGUYWY -- U=1 G=2 V=3 J=5 W=8 Y=9
XNNLNK + NXGLRX = RGKKLQ -- X=2 N=4 L=0 K=1 G=7 R=6 Q=3
0WDXL + 0CLLT = MRDXT -- W=3 D=4 X=5 L=0 C=9 T=6 M=1 R=2
0NYYG + 0YQJR = QNQNR -- N=2 Y=9 G=0 Q=1 J=3 R=4
QLX + QFG = JJF -- Q=0 L=2 X=3 F=9 G=6 J=1
Nota: a resposta pode ser diferente. A prova é dada pela aritmética.

Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

```
  WMPLZW
+ QMQJWQ
--------
 ZWWMWMU
```

Responda aqui:

valores das letras

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

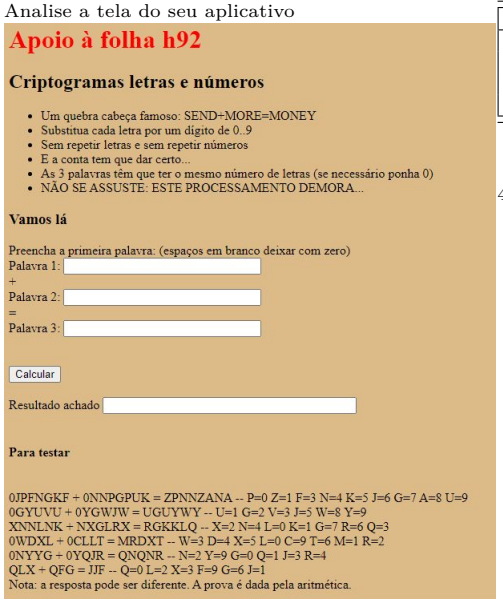
Depois que programar o aplicativo calcule a solução para o criptograma

```
  TAQEAE
+ TTAAJN
--------
 JJNZYEZ
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 16/09/2024 - 09:49:42.0 Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
Criptograma letras e números VIVXh92a V: 1.02
76087 LAURA PACHECO DELATTRE
24FRO408 - 9 _____ / _____ / _____

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

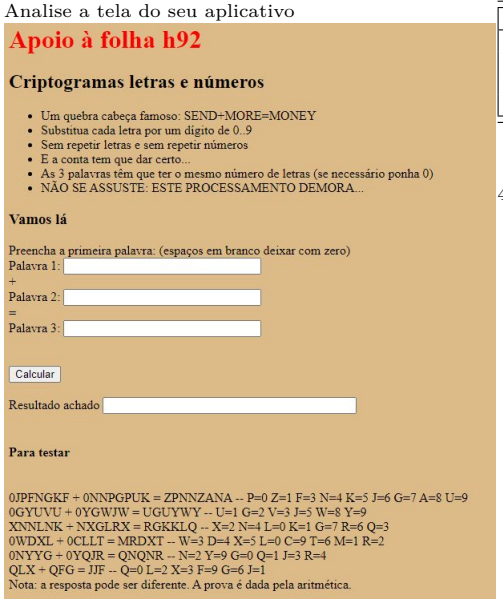
Depois que programar o aplicativo calcule a solução para o criptograma

```
  TNZJTH
+ JVMJZV
--------
 MRMNTQX
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



408-76087 - /

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

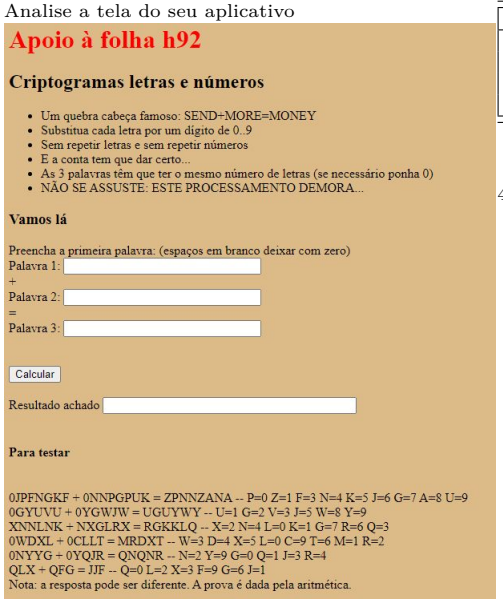
Depois que programar o aplicativo calcule a solução para o criptograma

```
  DNBVZQ
+ HAZHVD
--------
 HHVRHSN
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

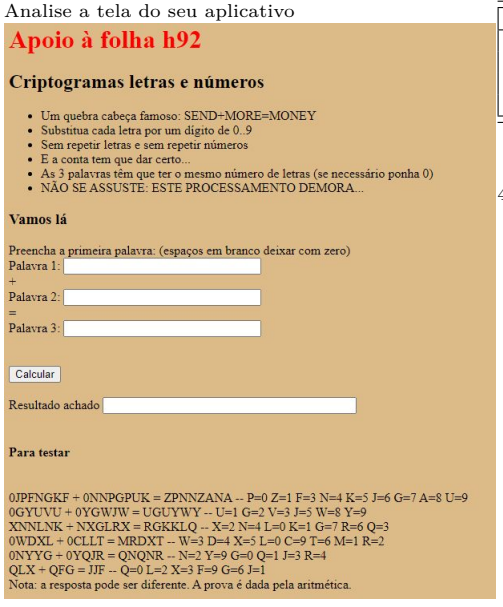
Depois que programar o aplicativo calcule a solução para o criptograma

```
  BBGFUM
+ SGGUUG
--------
 FUBGEMM
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 16/09/2024 - 09:49:42.0 Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
Criptograma letras e números VIVXh92a V: 1.02
76113 NICOLE DA SILVA DE AZEVEDO
24FRO408 - 12 _____ / _____ / _____

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY

Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta

Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade

É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

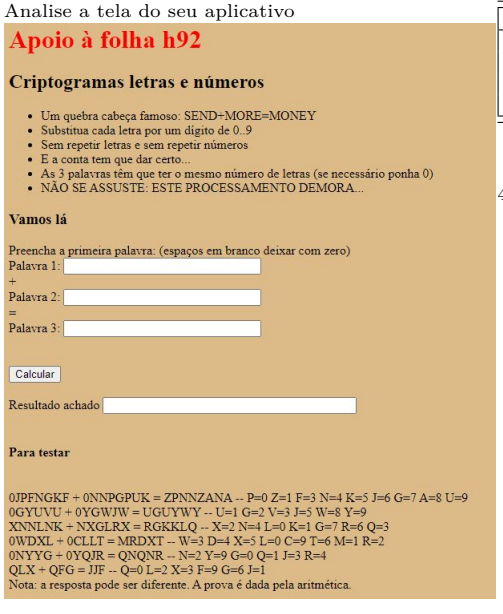
```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Analise a tela do seu aplicativo



Apoio à folha h92

Criptogramas letras e números

- Um quebra cabeça famoso: SEND+MORE=MONEY
- Substitua cada letra por um dígito de 0..9
- Sem repetir letras e sem repetir números
- E a conta tem que dar certo...
- As 3 palavras têm que ter o mesmo número de letras (se necessário ponha 0)
- NÃO SE ASSUSTE: ESTE PROCESSAMENTO DEMORA...

Vamos lá

Preencha a primeira palavra: (espaços em branco deixar com zero)
Palavra 1:
+
Palavra 2:
=
Palavra 3:

Calcular

Resultado achado

Para testar

0JPFNGKF + 0NNPGPUK = ZPNNZANA -- P=0 Z=1 F=3 N=4 K=5 J=6 G=7 A=8 U=9
0GYUVU + 0YGWJW = UGUYWY -- U=1 G=2 V=3 J=5 W=8 Y=9
XNNLNK + NXGLRX = RGKKLQ -- X=2 N=4 L=0 K=1 G=7 R=6 Q=3
0WDXL + 0CLLT = MRDXT -- W=3 D=4 X=5 L=0 C=9 T=6 M=1 R=2
0NYYG + 0YQJR = QNQNR -- N=2 Y=9 G=0 Q=1 J=3 R=4
QLX + QFG = JJF -- Q=0 L=2 X=3 F=9 G=6 J=1
Nota: a resposta pode ser diferente. A prova é dada pela aritmética.

Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

```
  JPWPJY
+ WGYVPJ
--------
 BGAJJVP
```

Responda aqui:

valores das letras

408-76113 - /

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

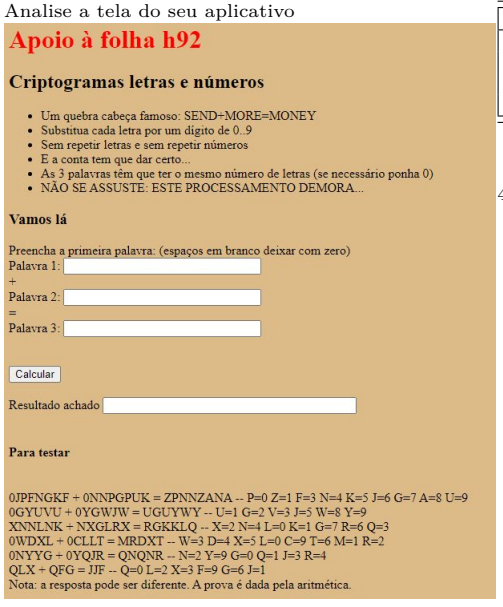
Depois que programar o aplicativo calcule a solução para o criptograma

```
  TFHEEW
+ BHWHYT
--------
 VYFWYFV
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

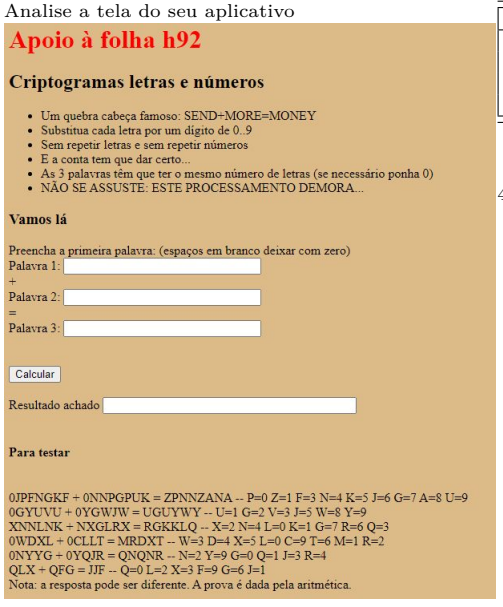
```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Analise a tela do seu aplicativo



Apoio à folha h92

Criptogramas letras e números

- Um quebra cabeça famoso: SEND+MORE=MONEY
- Substitua cada letra por um dígito de 0..9
- Sem repetir letras e sem repetir números
- E a conta tem que dar certo...
- As 3 palavras têm que ter o mesmo número de letras (se necessário ponha 0)
- NÃO SE ASSUSTE: ESTE PROCESSAMENTO DEMORA...

Vamos lá

Preencha a primeira palavra: (espaços em branco deixar com zero)
Palavra 1:
+
Palavra 2:
=
Palavra 3:

Calcular

Resultado achado

Para testar

0JPFNGKF + 0NNPGPUK = ZPNNZANA -- P=0 Z=1 F=3 N=4 K=5 J=6 G=7 A=8 U=9
0GYUVU + 0YGWJW = UGUYWY -- U=1 G=2 V=3 J=5 W=8 Y=9
XNNLNK + NXGLRX = RGKKLQ -- X=2 N=4 L=0 K=1 G=7 R=6 Q=3
0WDXL + 0CLLT = MRDXT -- W=3 D=4 X=5 L=0 C=9 T=6 M=1 R=2
0NYYG + 0YQJR = QNQNR -- N=2 Y=9 G=0 Q=1 J=3 R=4
QLX + QFG = JJF -- Q=0 L=2 X=3 F=9 G=6 J=1
Nota: a resposta pode ser diferente. A prova é dada pela aritmética.

Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

```
  FJGFJF
+ CRJXGX
--------
 MGCFFFF
```

Responda aqui:

valores das letras

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 16/09/2024 - 09:49:42.0
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
Criptograma letras e números VIVXh92a V: 1.02
76144 RAUL KNAPKI CUNHA
24FRO408 - 15 _____ / _____ / _____

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY

Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta

Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade

É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

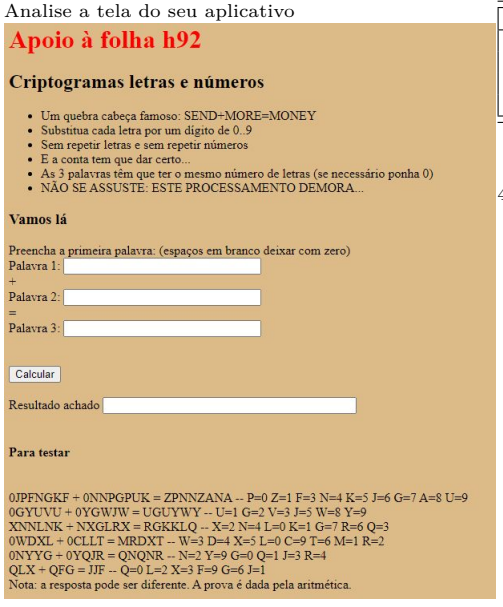
```
  ZGNRWK
+ PJZPWK
--------
  GWRJWP
```

Responda aqui:

valores das letras

408-76144 - /

Analise a tela do seu aplicativo



CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 16/09/2024 - 09:49:42.0 Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
Criptograma letras e números VIVXh92a V: 1.02
76168 VICTOR BRONDANI DOS SANTOS
24FRO408 - 16 _____ / _____ / _____

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

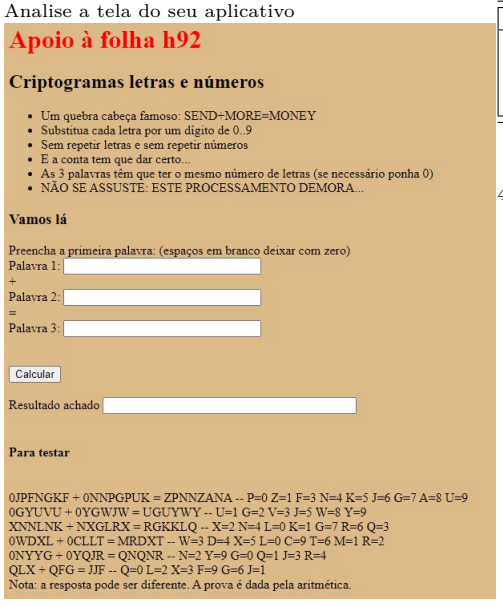
Depois que programar o aplicativo calcule a solução para o criptograma

```
  YPHMGM
+ GYPNGP
--------
 HGGNGNA
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



408-76168 - /

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

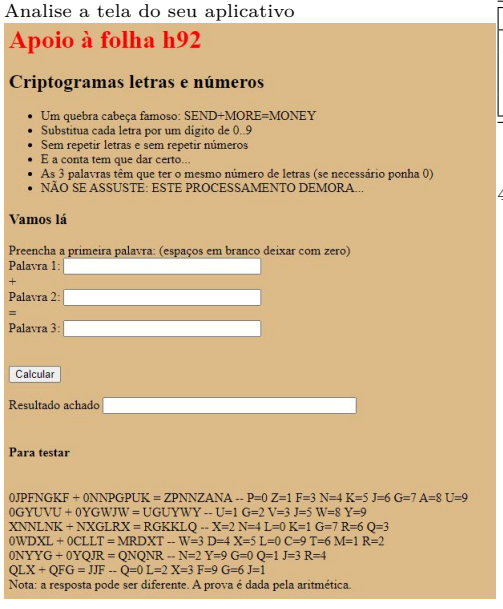
```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Analise a tela do seu aplicativo



Apoio à folha h92

Criptogramas letras e números

- Um quebra cabeça famoso: SEND+MORE=MONEY
- Substitua cada letra por um dígito de 0..9
- Sem repetir letras e sem repetir números
- E a conta tem que dar certo...
- As 3 palavras têm que ter o mesmo número de letras (se necessário ponha 0)
- NÃO SE ASSUSTE: ESTE PROCESSAMENTO DEMORA...

Vamos lá

Preencha a primeira palavra: (espaços em branco deixar com zero)
Palavra 1:
+
Palavra 2:
=
Palavra 3:

Calcular

Resultado achado

Para testar

0JPFNGKF + 0NNPGPUK = ZPNNZANA -- P=0 Z=1 F=3 N=4 K=5 J=6 G=7 A=8 U=9
0GYUVU + 0YGWJW = UGUYWY -- U=1 G=2 V=3 J=5 W=8 Y=9
XNNLNK + NXGLRX = RGKKLQ -- X=2 N=4 L=0 K=1 G=7 R=6 Q=3
0WDXL + 0CLLT = MRDXT -- W=3 D=4 X=5 L=0 C=9 T=6 M=1 R=2
0NYYG + 0YQJR = QNQNR -- N=2 Y=9 G=0 Q=1 J=3 R=4
QLX + QFG = JJF -- Q=0 L=2 X=3 F=9 G=6 J=1
Nota: a resposta pode ser diferente. A prova é dada pela aritmética.

Avaliação:

☞ Para você fazer

Depois que programar o aplicativo calcule a solução para o criptograma

```
  PAKPBM
+ MXKJMC
--------
  DKXXXJ
```

Responda aqui:

valores das letras

Criptograma letras e números

Mais um exercício para desenvolver as habilidades até agora trabalhadas, a saber: HTML, CSS e finalmente a programação Javascript. O aplicativo é para achar a resposta de um criptograma.

SEND + MORE = MONEY Existe um quebra-cabeça famoso, no qual a expressão SEND+MORE=MONEY deve ter cada letra substituída por um dígito numérico, sem repetir dígitos (isto é cada letra diferente tem um dígito diferente) e após a substituição a conta dá certo. Existem inúmeras respostas, mas se se fizer a ressalva de que $M \neq 0$, só há uma resposta que é $M = 1$, $O = 0$, $N = 6$, $D = 7$, $S = 9$, $E = 5$, $R = 8$, *e* $Y = 2$. Substitua e veja como a conta está certa.

Antes de prosseguir, vale ressaltar que esta é uma das funções importantes na computação que é fácil num sentido e muito difícil no outro. É o caso: passar de números a letras é trivial. Basta escolher dois números aleatórios, somá-los, escolher 10 letras a gosto e efetuar as substituições.

Já a volta, como veremos a seguir, demanda muitos (milhões, bilhões de ciclos de processamento). Aliás, não se espante se ao rodar este programa ele demorar vários minutos. Não é sempre que o computador resulta instantâneo...

Esta folha vai introduzir alguns conceitos bacanas, vamos lá:

Força bruta Vamos criar uma função que vai alocando dígitos às letras, e a cada alocação ela verifica se a conta deu certo. Se deu, aleluia, o problema acabou. Se não deu, vamos para a próxima alternativa. Note que esta estratégia só pode ser dada a um computador. Um ser humano que resolvesse o problema assim, demoraria anos para chegar a um resultado. Testam-se TODAS as possibilidades até a certa aparecer.

Recursividade É um complemento da técnica anterior. É sempre o mesmo programa que é chamado. Ou seja, você programa uma única solução e o mesmo programa se auto-chama bilhões de vezes, até a solução certa aparecer. Se você não quiser usar a recursividade, terá que controlar todos os detalhes que manipulam e verificam cada um dos testes. Acredite em mim, é muito mais fácil usar a recursividade.

Exemplos destas duas técnicas usadas junto: o passeio do cavalo no tabuleiro de xadrez, a solução de qualquer sudoku, a divisão ao meio de alguma coleção de objetos, etc etc.

Para resolver este problema, o programa vai usar as 3 palavras digitadas (Sempre em letras maiúsculas, sem espaços, com eventuais zeros na frente para garantir tamanhos iguais nas 3 palavras). Note que o programa — para ser mais simples — não testa isso tudo. Então, se você cometer um erro de digitação o programa vai fazer coisas erradas/inesperadas.

O programa vai montar uma matriz de 3 linhas por n colunas e a cada tentativa de substituir letras por números, vai recriar uma matriz de mesma dimensão e vai testar a linha[1]+linha[2] = linha[3] ? Se esta condição der certo o criptograma está resolvido, senão toca a procurar outra combinação.

Truques JS que vale a pena aprender

Índice de um caracter em string Para descobrir usa-se `i=nomestring.indexOf('carac')` onde `carac` é o caractere que se está procurando, `nomestring` é a variável string e `i` é o índice da primeira ocorrência do caractere no string.

Linhas e colunas em matriz Para saber o número de linhas é `mat.length` e o número de colunas é `mat[0].length`.

Converter linha de matriz em inteiro Supondo que a linha da matriz só contenha digitos, o comando que faz isso é `linha=mat[i].join('')` seguido de `numero=parseInt(linha)`.

Criar matriz com palavras A ideia é criar uma matriz de 3 linhas e n colunas, a depender do tamanho que foi digitado.

```
let conta=[]
conta.push(p1.split(''))
conta.push(p2.split(''))
conta.push(p3.split(''))
```

Como tirar letras repetidas Esta é uma estratégia clássica. Basta converter tudo em um conjunto (que lembrando, não aceita elementos repetidos) e depois desconverter:

```
letras=new Set(p1+p2+p3)
letras.delete('0')
letrasString = [...letras].join('');
```

Como criar uma matriz de caracteres para criar uma matriz de caracteres

```
let m=Array.from({length : conta.length},
() => Array(conta[0].length).fill (' '));
```

let variável O JS tem umas questões de escopo que fogem um pouco ao trivial de outras linguagens. Os comandos

```
let quais = menos('0123456789', ja)
let ii
```

abaixo, se ausentes, bagunçam completamente as chamadas recursivas.

o JS

```
<script type="text/javascript">
function troca(a){
  let m=Array.from({length : conta.length},
  () => Array(conta[0].length).fill (' '));
  for (i=0;i<m.length;i++){
   for (j=0;j<m[0].length;j++){
    if (conta[i][j]=='0') {m[i][j]='0'}
    else {
     ca=conta[i][j]
     k=a.indexOf(ca)
     m[i][j]=a[k+1]
   }
  }
 }
 if ((parseInt(m[0].join(''))+parseInt(m[1].
  join(''))==parseInt(m[2].join('')))
    {return true}
  else {return false;}
}
function menos(aa,bb){
 rr=''
 for(zica=0;zica<aa.length;zica++){
  if (!(bb.includes(aa[zica]))) {rr=rr+aa[zica]}
 }
 return rr
}
function sol(ja,aindan){
 if (ACABOU==1){return}
 if (aindan.length==0){
   if (troca(ja)) {RESULTADO=ja;ACABOU=1;
     return true}
   else {
     return false}
   }
   let quais = menos('0123456789', ja)
   let ii
   for(ii=0;ii<quais.length;ii++){
     sol(ja+aindan[0]+quais[ii],aindan.slice(1));
   }
  return true
}
let conta=[]
let RESULTADO
let ACABOU=0
function procure(){
    p1=document.getElementById('p1').value;
    p2=document.getElementById('p2').value;
    p3=document.getElementById('p3').value;
    letras=new Set(p1+p2+p3)
 letras.delete('0')
 letrasString = [...letras].join('');
 conta.push(p1.split(''))
 conta.push(p2.split(''))
 conta.push(p3.split(''))
 sol('',letrasString)
 document.getElementById('res').value=RESULTADO
 }
</script>
```

Avaliação:

☞ Para você fazer

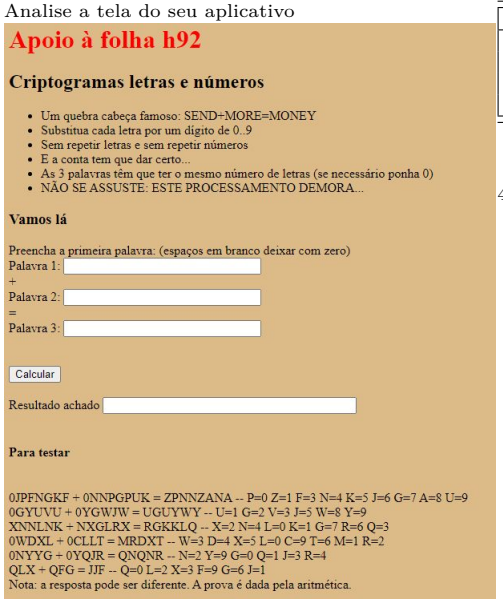
Depois que programar o aplicativo calcule a solução para o criptograma

```
  BFEZEY
+ YQBUNB
--------
 GHYNQHQ
```

Responda aqui:

valores das letras

Analise a tela do seu aplicativo



Apoio à folha h92

Criptogramas letras e números

- Um quebra cabeça famoso: SEND+MORE=MONEY
- Substitua cada letra por um dígito de 0..9
- Sem repetir letras e sem repetir números
- E a conta tem que dar certo...
- As 3 palavras têm que ter o mesmo número de letras (se necessário ponha 0)
- NÃO SE ASSUSTE: ESTE PROCESSAMENTO DEMORA...

Vamos lá

Preencha a primeira palavra: (espaços em branco deixar com zero)
Palavra 1:
+
Palavra 2:
=
Palavra 3:

Calcular

Resultado achado

Para testar

0JPFNGKF + 0NNPGPUK = ZPNNZANA -- P=0 Z=1 F=3 N=4 K=5 J=6 G=7 A=8 U=9
0GYUVU + 0YGWJW = UGUYWY -- U=1 G=2 V=3 J=5 W=8 Y=9
XNNLNK + NXGLRX = RGKKLQ -- X=2 N=4 L=0 K=1 G=7 R=6 Q=3
0WDXL + 0CLLT = MRDXT -- W=3 D=4 X=5 L=0 C=9 T=6 M=1 R=2
0NYYG + 0YQJR = QNQNR -- N=2 Y=9 G=0 Q=1 J=3 R=4
QLX + QFG = JJF -- Q=0 L=2 X=3 F=9 G=6 J=1
Nota: a resposta pode ser diferente. A prova é dada pela aritmética.