Licenciatura em Matemática

Introdução à Computação

Prof. Roberto Hirata Jr. e Walter F. Mascarenhas

Correção da segunda prova aplicada em 20/12/2004

Q1.
```
void f0(int V[], int n, int k) {
  int i ;
  for (i=0; i<n; i=i+1)
    V[i] = V[i] + k ;
}

void f1(int V[], int n) {
  int i = 0 ;
  while(((V[i]%3)!=0)&&(i<n)) {
    V[i] = V[i]%2 ;
       i = i+1 ;
  }
}

void f2(int V[], int n) {
  int i ;
  for (i=1; i<n; i=i+1)
    V[i] = V[i-1] + V[i] ;
}

int main() {
  int nusp, V[9], ndigitos = 0 ;

  printf(``Digite o seu número USP\n'') ;
  scanf(``\%d'',&nusp) ;
  /* Azar do usuário se ele digitou errado */
  while (nusp!=0) {
    V[ndigitos] = nusp%10 ;
    nusp = nusp/10 ;
    ndigitos = ndigitos + 1 ;
  }
  for (i = 1; i < ndigitos; i = i+1)
    printf(``%d '',V[i-1]) ;
  printf(``%d\n'',V[ndigitos-1]) ;

  f0(V,ndigitos,10) ;
  for (i = 1; i < ndigitos; i = i+1)
```

```
    printf(``%d '',V[i-1]) ;
  printf(``%d\n'',V[ndigitos-1]) ;

  f2(V,ndigitos) ;
  for (i = 1; i < ndigitos; i = i+1)
    printf(``%d '',V[i-1]) ;
  printf(``%d\n'',V[ndigitos-1]) ;

  f1(V,ndigitos) ;
  for (i = 1; i < ndigitos; i = i+1)
    printf(``%d '',V[i-1]) ;
  printf(``%d\n'',V[ndigitos-1]) ;

  return(0) ;
}
```

passo	ndigitos	V[0]	V[1]	V[2]	V[3]	V[4]	V[5]	V[6]
1	0							
2		3						
3	1							
4			4					
5	2							
6				1				
7	3							
8					9			
9	4							
10						6		
11	5							
12							8	
13	6							
14								2
15	7							
16		3	4	1	9	6	8	2
17		13	14	11	19	16	18	12
18		13	27	38	57	73	91	103
19		1	27	38	57	73	91	103

Q2. Escreva uma função `int VetorLecker(int V[], int n)` cuja especificação está descrita a seguir: dizemos que uma seqüência de números inteiros, **sem elementos repetidos**, é **lecker** se ela tem **apenas** um elemento que é maior que seus vizinhos. Por exemplo,

i) 2 5 10 **45** 23 11 7 é lecker pois 45 é o único elemento que é maior que seus vizinhos que são o 10 e o 23.

ii) **13** 5 4 2 **3** 0 -3 -5 **não é** lecker pois há dois números que são maiores que seus vizinhos, o 13 e o 3.

iii) 1 2 3 **4** é **lecker** pois 4 é o único número maior que seus vizinhos (3 é o único vizinho do 4).

Escreva a função `VetorLecker` que recebe como parâmetros um vetor `V` e seu tamanho `n` e retorna -1 se o vetor não for lecker, ou o índice do elemento que faz com que o vetor seja lecker.

```
int VetorLecker(int V[], int n) {
  int i, res, cont, limite ;

  cont = 0 ;
  limite = n-1 ;

  /* Verifica o primeiro elemento do vetor */
  if (V[0] > V[1]) {
    res = 0 ;
    cont = cont + 1 ;
  }
  /* Verifica os elementos do meio do vetor */
  for (i=1; i<limite; i++) {
    if ((V[i]>V[i-1])&&(V[i]>V[i+1])) {
      res = i ;
      cont = cont + 1 ;
    }
  }
  /* Verifica o último elemento do vetor */
  if (V[limite] > V[limite-1]) {
    res = limite ;
    cont = cont + 1 ;
  }
  /* O vetor V não tem elementos repetidos, assim,
```

```
     cont é 1, ou maior que 1. Por quê?            */
  if (cont > 1) {
    res = -1 ;
  }
  return(res) ;
}
```

Q3. Escreva uma função `float rangeSum(float V[], int n, int ind, int k)`, $0 \leq$ `ind` $<$ `n`, $0 <$ `k` $<$ `n`, cuja especificação encontra-se abaixo.

$$\sum_{i=\texttt{ind}\breve{-}\texttt{k}/2}^{\texttt{ind}\breve{+}\texttt{k}/2} V[i]$$

onde, dados $a, b \geq 0$, $a \breve{-} b = 0$ se $a \leq b$, ou $a - b$ caso contrário; e $a \breve{+} b = n - 1$ se $a + b \geq n - 1$, ou $a + b$ caso contrário.

Escreva a função `rangeSum` que recebe como parâmetros um vetor `V`, seu tamanho `n` e dois inteiros não-negativos, `ind` e `k` e retorna o valor da soma acima.

```
float rangeSum(float V[], int n, int ind, int k) {
  int i, ks2, limInferior, limSuperior ;
  float res = 0 ;

  ks2 = k/2 ;

  if (ind<=ks2) {
    limInferior = 0 ;
  }
  else limInferior = ind - ks2 ;

  if ((ind+ks2)>=(n-1)) {
    limSuperior = n-1 ;
  }
  else limSuperior = ind + ks2 ;

  for(i=limInferior; i<=limSuperior; i++) {
    res = res + V[i] ;
  }

  return(res) ;
}
```