MAC2166 – Introdução à Computação para Engenharia

Escola Politécnica

Terceira Prova – 22 de junho de 2009

Nome: _____

Assinatura: _____

Nº USP: _____ Turma: _____

Professor: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. A prova consta de 3 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
3. A prova pode ser feita a lápis. Cuidado com a legibilidade.
4. Qualquer questão pode ser resolvida em qualquer página. Se a questão não está na página correspondente ao enunciado basta indicar isto na página e escrever QUESTÃO X em letras ENORMES antes da solução.
5. Não é necessário apagar rascunhos no caderno de questões.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Não é permitido o uso de calculadoras.
8. Não é permitido a consulta a livros, apontamentos ou colegas.

DURAÇÃO DA PROVA: 2 horas

Questão	Valor	Nota
1	2,0	
2	3,0	
3(a)	1,0	
3(b)	2,0	
3(c)	2,0	
Total	10,0	

1. (vale 2 pontos)

   Simule a execução do programa abaixo, destacando a sua saída. A saída do programa consiste de tudo que resulta dos comandos `printf`.

```
#include <stdio.h>

int f0(int nusp, int V[10]) {
  int n=0 ;

  while(nusp!=0) {
    V[n] = nusp%10 ;
    n++;
    nusp=nusp/10 ;
  }
  return n;
}

void f1(int *a, int *b) {
  int aux ;

  aux = *a ;
  *a = *b ;
  *b = aux ;
}

int f2(int *n, int V[10]) {
  int i, e, d ;

  V[*n]=0; V[*n+1]=1;
  *n = *n+2 ;
  for(i=0; i<*n; i++) printf("%d ", V[i]) ;
  printf("\n");

  e=0; d=*n-1;
  while (e < d) {
    while(e<*n && V[e]%2==1) e++;
    while(d>=0 && V[d]%2==0) d--;
    if(e<d) {
       f1(&V[e],&V[d]) ;
    }
    printf("e=%d d=%d\n", e, d) ;
  }
  return e;
}
```

```
int f3(int n, int V[10], int t) {
  int i ;

  for(i=1;i<t;i++) {
      V[i] = (V[i-1]+V[i]) % 2;
  }
  for(i=t+1; i<n; i++) {
      V[i] = (V[i-1]+V[i]) % 2;
  }

  return(V[t]) ;
}


int main() {
  int nusp, i, n, t, V[10] ;

  printf("Digite o seu NUSP: ") ;
  scanf("%d", &nusp) ;

  n = f0(nusp, V) ;
  printf("nusp=%d n=%d\n", nusp, n) ;
  for(i=0; i<n; i++) printf("%d ", V[i]) ;
  printf("\n");

  t = f2(&n,V) ;
  printf("n=%d t=%d\n", n, t) ;
  for(i=0; i<n; i++) printf("%d ", V[i]) ;
  printf("\n");

  t = f3(n,V,t) ;
  for(i=0; i<n; i++) printf("%d ", V[i]) ;
  printf("\n");
  printf("t=%d\n", t) ;

  return 0;
}
```

Para efeito de correção só será considerada a saída do programa. Você pode usar o espaço livre abaixo como bem entender.

Saída do programa

2. (vale 3,0 pontos)

Deseja-se escrever uma função `void roda(int v[MAX], int n, ind d)` que rode circularmente um vetor `v`, de tamanho `n`, por `d` posições à direita; a implementação deve funcionar para `0 < d < n <= MAX`, e não importa o que ocorre com outros valores de `d` e `n`. Para entender esse "vetor circular", pense nos seus elementos como escritos ao longo de um círculo, como num mostrador de relógio analógico. Ao girar por `d` posições no sentido horário, os elementos se deslocam, e os que estavam no fim vão parar no começo, conforme o exemplo abaixo.

Exemplo: Para `v = [4 3 1 2 6 5]` e `n` = 6, temos:

`d`	`v` depois de `roda(v, n, d)`
1	5 4 3 1 2 6
2	6 5 4 3 1 2
3	2 6 5 4 3 1
4	1 2 6 5 4 3
5	3 1 2 6 5 4

Um caso particular curioso é quando `n` = 2; neste caso, o único valor possível para `d` é `d` = 1, e neste caso os dois elementos do vetor trocam de posição. Foram escritas várias propostas de soluções. **Para cada função sugerida na próxima página, indique no quadro correspondente se está correta ou não, e, caso não esteja correta, dê um exemplo, mostrando o que a função deveria devolver e o que devolve.**

```
void tic(int v[MAX], int n){
   int i, t;
   t = v[n-1];
   for(i = n - 1; i > 0; i--)
       v[i] = v[i-1];
   v[0] = t;
}
void roda(int v[MAX], int n, int d) {
   while(d > 0) {
       tic(v, n);
       d-- ;
   }
}
```

☐ Funciona sempre.

☐ Falha para `d` = ☐,
`v` = ☐,
pois devolve ☐,
mas deveria devolver ☐.

```
void roda(int v[MAX], int n, int d) {
   int i, t, q;
   q = v[0];
   for(i = 1; i < n; i++) {
       t = v[i];
       v[i] = q;
       q = v[(i+d) % n];
   }
   v[0] = t;
}
```

☐ Funciona sempre.

☐ Falha para `d` = ☐,
`v` = ☐,
pois devolve ☐,
mas deveria devolver ☐.

```
void tac(int v[MAX], int n){
   int i, t;
   t = v[0];
   for(i = 0; i < n - 1; i++)
       v[i] = v[i + 1];
   }
   v[i] = t;
}
void roda(int v[MAX], int n, int d) {
   for(d = n - d; d > 0; d--)
       tac(v, n);
}
```

☐ Funciona sempre.

☐ Falha para `d` = ☐,
`v` = ☐,
pois devolve ☐,
mas deveria devolver ☐.

```
void roda(int v[MAX], int n, int d) {
   int i, t;
   t = v[n-d];
   for(i = n - 1; i > 0; i = i-d) {
       if ( i < d ) i = i+n;
       v[i] = v[i - d];
   }
   v[0] = t;
}
```

☐ Funciona sempre.

☐ Falha para `d` = ☐,
`v` = ☐,
pois devolve ☐,
mas deveria devolver ☐.

```
void roda(int v[MAX], int n, int d) {
   int aux[MAX], i;
   for(i = 0; i < d; i++)
      aux[i] = v[n - d + i];
   while(i < n) {
      v[i] = v[i - d];
      i++ ;
   }
   for(i = 0; i < d; i++)
      v[i] = aux[i];
}
```

☐ Funciona sempre.

☐ Falha para `d` = ☐,
`v` = ☐,
pois devolve ☐,
mas deveria devolver ☐.

3. (vale 5,0 pontos)

No EP4, um dos filtros que você implementou foi o filtro da média. Aqui você irá escrever três funções que implementam o **filtro da média ponderada**. Ele é similar ao filtro da média, exceto pelo fato de que a média calculada é uma média ponderada. Lembre que para calcular a média ponderada de uma coleção de números, multiplicamos cada valor por seu peso, somamos tudo e dividimos pela soma dos pesos. A divisão a ser usada é divisão inteira. Usaremos dois sistemas de pesos, definidos adiante.

Esta questão está dividida em 3 itens. **Você pode utilizar funções auxiliares se desejar, mas DEVE ESCREVER todas elas**; isto é, não use funções da biblioteca do C (por exemplo, da `math.h`).

Assuma que as constantes `MAXLINHA` e `MAXCOLUNA` já estão devidamente definidas.

a) (vale 1,0 ponto) Escreva uma função com protótipo

```
int peso(int x, int y, int i, int j, int Ponderacao);
```

que recebe as coordenadas $(\texttt{x}, \texttt{y})$ de um ponto na imagem e as coordenadas $(\texttt{i}, \texttt{j})$ de um ponto na vizinhança de $(\texttt{x}, \texttt{y})$, e devolve o peso associado à coordenada $(\texttt{i}, \texttt{j})$, segundo a fórmula:

se $\texttt{Ponderacao} = 1$, devolve $|\texttt{i} - \texttt{x}| + |\texttt{j} - \texttt{y}|$
senão devolve $|\texttt{i} + \texttt{j} - \texttt{x} - \texttt{y}|$

onde $|\texttt{a}|$ é o valor absoluto (módulo) do inteiro `a`. Ou seja, `Ponderacao` é uma forma de escolher entre os dois sistemas de pesos.

b) (vale 2,0 pontos)
Escreva uma função com protótipo

```
int valorMedio(int imagem[MAXLINHA][MAXCOLUNA], int linhas, int colunas,
                int larguraJanela, int x, int y, int Ponderacao);
```

Os 4 primeiros parâmetros são como no EP4. Só para lembrar:

- `imagem` é a matriz de trabalho;
- `linhas` e `colunas` definem as dimensões da imagem;
- `larguraJanela` é um número ímpar que define a janela em torno de cada ponto da imagem que será usada para calcular o novo valor do ponto.

Os outros parâmetros são

- `x`, `y` são as coordenadas de um ponto da imagem (`0` $<=$ `x` $<$ `linhas` e `0` $<=$ `y` $<$ `colunas`);
- `Ponderacao` é um parâmetro com valor 1 ou 2, que escolhe um peso conforme o item **(a)**.

A função `valorMedio()` devolve o valor médio dos pontos da imagem na vizinhança do ponto de coordenadas (`x`, `y`) de acordo com a média ponderada onde o peso de um ponto é dado pela função do item **(a)**. A vizinhança a ser considerada é a definida pela janela com largura `larguraJanela` e devem ser considerados apenas os pontos na vizinhança com coordenadas válidas. Use a função `peso`, mesmo que não a tenha escrito.

c) (vale 2,0 pontos) Escreva uma função com o protótipo

```
void filtroMedia(int imagem[MAXLINHA][MAXCOLUNA], int linhas, int colunas,
                 int larguraJanela, int Ponderacao);
```

Os parâmetros (exceto `x`, `y`, que aqui não aparecem) são como no item anterior.

Esta função deve aplicar a média ponderada a cada ponto da imagem, usando a vizinhança dada pela janela com largura `larguraJanela` e com o peso escolhido por `Ponderacao`. Use a função `valorMedio()` do item anterior, mesmo que não a tenha escrito.