MAC2166 – Introdução à Computação para Engenharia

Escola Politécnica

Terceira Prova – 25 de junho de 2012

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 equipamentos eletrônicos.
8. Não é permitido a consulta a livros, apontamentos ou colegas.

DURAÇÃO DA PROVA: 2 horas

Questão	Valor	Nota
1	2,5	
2(a)	1,0	
2(b)	1,5	
2(c)	1,5	
3(a)	1,5	
3(b)	2,0	
Total	10,0	

QUESTÃO 1 (vale 2,5 pontos)

Simule a execução do programa abaixo, destacando a sua saída. A saída do programa consiste de tudo que resulta dos comandos `printf`. **Para efeito de correção só será considerada a saída do programa**.

```
#include <stdio.h>

#define MAX 2

void f1(int a[MAX][MAX], int *d, int *c);
int  f2(int v[MAX], int w[MAX],
        int x, int *y);

int main() {
  int nusp, a[MAX][MAX], c, ind, v[MAX+1];

  printf("Digite o seu No. USP: ");
  scanf("%d",&nusp);
  ind = nusp%3;
  printf ("0: %d %d\n",nusp,ind);

  v[0]=13; v[1]=17; v[2]=9;
  a[0][0]=v[ind]%2;   a[0][1]=v[ind]%2+2;
  a[1][0]=v[ind]%3+4; a[1][1]=v[ind]%3+7;
  c=v[ind]%5+10;

  printf("7: %d %d %d\n",v[0],v[1],v[2]);
  printf("8: %d %d\n",a[0][0],a[0][1]);
  printf("9: %d %d\n",a[1][0],a[1][1]);
  printf("2: %d\n",c);

  f1(a,&c,&a[1][0]);

  printf("4: %d %d\n",a[0][0],a[0][1]);
  printf("5: %d %d\n",a[1][0],a[1][1]);
  printf("6: %d\n",c);

  v[0]=13; v[1]=17; v[2]=9;
  a[0][0]=v[ind]%2;   a[0][1]=v[ind]%2+2;
  a[1][0]=v[ind]%3+4; a[1][1]=v[ind]%3+7;

  a[1][1] = f2(a[0],a[1],a[0][1],&a[1][1]);

  printf("16: %d %d\n",a[0][0],a[0][1]);
  printf("17: %d %d\n",a[1][0],a[1][1]);

  return 0;
}

void f1(int a[MAX][MAX], int *d, int *c) {

  printf("1: %d %d\n",*d,*c);
  a[1][0] = *d; *d=a[0][1];
  *c=a[1][1]; a[0][0]=60;

}

int f2(int v[MAX], int w[MAX], int x, int *y) {

  printf("12: %d %d\n",v[0],w[0]);
  v[0]=w[0]; w[0]=x; x=22; *y=30; v[1]=w[1];
  printf("13: %d\n",*y);
  return v[0]+w[1];

}
```

Escreva a saída do programa no espaço abaixo. As demais áreas brancas podem ser usadas livremente para fazer a simulação.

A solução depende do resto da divisão do número USP por 3 `nusp%3`.

Saída do programa

```
Digite o seu No. USP: "nusp%3 == 0"
0: 12 0
7: 13 17 9
8: 1 3
9: 5 8
2: 13
1: 13 5
4: 60 3
5: 8 8
6: 3
12: 1 5
13: 30
16: 5 30
17: 3 35


Digite o seu No. USP: "nusp%3 == 1"
0: 1 1
7: 13 17 9
8: 1 3
9: 6 9
2: 12
1: 12 6
4: 60 3
5: 9 9
6: 3
12: 1 6
13: 30
16: 6 30
17: 3 36


Digite o seu No. USP: "nusp%3 == 2"
0: 2 2
7: 13 17 9
8: 1 3
9: 4 7
2: 14
1: 14 4
4: 60 3
5: 7 7
6: 3
12: 1 4
13: 30
16: 4 30
17: 3 34
```

QUESTÃO 2 (vale 4,0 pontos)

ITEM 2(a) (vale 1,0 ponto) Escreva uma função de protótipo

```
void diferenca(float x[MAX], float y[MAX], float diff[MAX], int d) ;
```

que recebe três vetores reais `x`, `y` e `diff` e um inteiro `d` e calcula a diferenca $\texttt{x} - \texttt{y}$ entre os vetores `x` e `y` em `diff`. O número de elementos nos vetores é `d`.

Exemplo: se $\texttt{x} = (5.1, 9.3, 2.2)$ e $\texttt{y} = (2.0, 4.2, 1.2)$, então $\texttt{diff} = (3.1, 5.1, 1.0)$.

```
/*
 *  Solucao
 */
void diferenca(float x[MAX], float y[MAX], float diff[MAX], int d)
{
  int i;

  for (i = 0; i < d; i++)
    diff[i] = x[i] - y[i];

}
```

ITEM 2(b) (vale 1,5 ponto) Escreva uma função de protótipo

```
float norma(float x[MAX], int d, float p) ;
```

que recebe um vetor real `x`, um inteiro `d`, e um real `p` e devolve a `p`-norma do vetor `x` de dimensão `d` (i.e., $||\mathtt{x}||_\mathtt{p}$). A `p`-norma de um vetor `x` é dada pela fórmula:

$$||\mathtt{x}||_\mathtt{p} = (\sum_{i=0}^{\mathtt{d}-1} |\mathtt{x}_i|^\mathtt{p})^{1/\mathtt{p}},$$

onde $\mathtt{x}_i$ é o elemento `i` do vetor `x`.

Para esta questão, você deve usar **OBRIGATORIAMETE** as funcões (considere-as como dadas, ou seja, você não precisa escrevê-las) com protótipos

```
float modulo(float y) ;
float potencia(float y, float p) ;
```

A primeira função recebe um real `y` e devolve $|\mathtt{y}|$. A segunda função recebe dois reais `y` e `p` e devolve o resultado $\mathtt{y}^\mathtt{p}$.

Exemplos:

- se $\mathtt{x} = (-2.0, -1.0, 2.0)$, então $||\mathtt{x}||_2 = (|-2.0|^2 + |-1.0|^2 + |2.0|^2)^{1/2} = (4.0 + 1.0 + 4.0) = 9^{1/2} = 3$.
- se $\mathtt{x} = (-6.0, 8.0, -1.0)$, então $||\mathtt{x}||_3 = (|-6.0|^3 + |8.0|^3 + |-1.0|^3)^{1/3} = (216.0 + 512.0 + 1.0)^{1/3} = 729.0^{1/3} = 9.0$.
- se $\mathtt{x} = (-5.0, 1.0, -6.0)$, então $||\mathtt{x}||_1 = (|-5.0| + |1.0| + |-6.0|)^1 = (5.0 + 1.0 + 6.0)^1 = 12.0$.

```
/*
 * Solucao 1
 */
float norma(float x[MAX], int d, float p)
{
  int i;
  float moduloxi;
  float soma;
  float pnorma;

  soma = 0;
  for (i = 0; i < d; i++)
    {
      modulox = modulo(x[i]);
      xp      = potencia(modulox,p);
      soma    = soma + xp;
    }

  pnorma = potencia(soma,1/p);

  return pnorma;
}
```

```
/*
 * Solucao 2
 */
float norma(float x[MAX], int d, float p)
{
  int i;
  float soma;

  soma = 0;
  for (i = 0; i < d; i++)
    soma += potencia(modulo(x[i]),p);

  return potencia(soma,1/p);
}
```

ITEM 2(c) (vale 1,5 ponto)

Escreva um programa em C que lê **do teclado** dois inteiros `n` e `d` que são, respectivamente, o número de pontos e o número de dimensões de cada ponto; um real positivo `p`, que é a ordem da norma a ser calculada; e `n` pontos de dimensão `d` em forma de uma matriz `A` de dimensão `n` $\times$ `d`. Seu programa deve imprimir, dentre os `n` $-$ 1 primeiros pontos, um ponto cuja distância (`p`-norma) seja mais próxima do último ponto, bem como a sua distância `p`-norma.

A distância `p`-norma entre dois pontos `x` e `y` é $||\mathtt{x} - \mathtt{y}||_{\mathtt{p}}$.

Exemplo: se `n` $= 5$, `d` $= 3$ e `p` $= 1$, então o usuário deverá entrar com `n` $= 5$ pontos de dimensão `d` $= 3$, digamos, $(0.0, 1.0, 0.0)$, $(1.0, 0.0, -1.0)$, $(0.0, 2.0, 1.0)$, $(1.0, 2.0, 0.0)$, $(2.0, 1.0, 0.0)$ formando uma matriz $A_{n \times d}$:

$$\mathtt{A} = \begin{pmatrix} 0.0 & 1.0 & 0.0 \\ 1.0 & 0.0 & -1.0 \\ 0.0 & 2.0 & 1.0 \\ 1.0 & 2.0 & 0.0 \\ 2.0 & 1.0 & 0.0 \end{pmatrix}_{5 \times 3}$$

Neste caso, dos `n` $-$ 1 primeiros pontos, um ponto mais próximo do último $(2.0, 1.0, 0.0)$ (considerando a 1-norma) é $(0.0, 1.0, 0.0)$ com distância 1-norma de 2.0. Note que o ponto $(1.0, 2.0, 0.0)$ também é outro ponto mais próximo. Seu programa pode imprimir qualquer um deles.

Use **OBRIGATORIAMENTE** as funções dos itens anteriores, mesmo que você não as tenha feito.

```
#define MAX 256

/*
 * Solucao 1. A solucao supoe n >= 1 e d >= 1 e p != 0.
 */
int main()
{
  int   n;           /* numero de pontos   */
  int   d;           /* dimensao do espaco */
  float p;           /* ordem da norma */
  float A[MAX][MAX]; /* matriz com os pontos */

  float dist;        /* distancia de um ponto ao ponto origem */
  float mindist;     /* menor distancia de um ponto ao ponto origem */
  float imindist;    /* indice do ponto de menor distancia ate origem */

  float origem[MAX]; /* coodenada do ponto origem */
  float diff  [MAX];
  float x     [MAX];
  int i;
  int j;

  printf("Digite o numero de pontos e a dimensao: ");
  scanf("%d %d", &n, &d);

  printf("Digite a ordem da norma: ");
  scanf("%f", &p);
```

```
  /* leitura dos pontos */
  for (i = 0; i < n; i++)
    {
      printf("Digite o %do. ponto: ");
      for (j = 0; j < d; j++)
        {
          scanf("%f", &A[i][j]);
        }
    }

  /* copie origem e primeiro ponto */
  for (j = 0; j < d; j++)
    {
      origem[j] = A[n-1][j];
      x[j]      = A[0][j];
    }

  /* ponto 0 e' o de menor distancia ate que se prove o contrario*/
  diferenca(x,origem,diff,d);
  mindist  = norma(diff,d,p);
  imindist = 0;

  for (i = 1; i < n-1 ; i++)
    {
      /* copie ponto i para vetor x */
      for (j = 0; j < d; j++)
        {
          x[j] = A[i][j];
        }

      /* calcule a distancia entre x e a origem */
      diferenca(x,origem,diff,d);
      dist = norma(diff,d,p);

      /* verifique se o ponto i esta mais proximo da origem */
      if (dist < mindist)
        {
          mindist  = dist;
          imindist = i;
        }
    }

   printf("O ponto mais proximo do ponto origem e' = (");
   for (j = 0; j < d-1; j++)
     {
       printf("%f,", A[imindist][j]);
     }
   printf("%f)\n cuja distancia e' = %f\n", A[imindist][d-1], mindist);
   return 0;
}
```

```
/*
 * Solucao 1. A solucao supoe n >= 1 e d >= 1 e p != 0.
 *   Usa o fato de uma linha de matriz ter o comportamento de
 *   um vetor. Foram removidos algumas chaves superfluas.
 */
int main()
{
  int   n;           /* numero de pontos   */
  int   d;           /* dimensao do espaco */
  float p;           /* ordem da norma */
  float A[MAX][MAX]; /* matriz com os pontos */

  float dist;        /* distancia de um ponto ao ponto origem */
  float mindist;     /* menor distancia de um ponto ao ponto origem */
  float imindist;    /* indice do ponto de menor distancia ate origem */

  float origem[MAX]; /* coodenada do ponto origem */
  float diff  [MAX];
  int i;
  int j;

  printf("Digite o numero de pontos e a dimensao: ");
  scanf("%d %d", &n, &d);

  printf("Digite a ordem da norma: ");
  scanf("%f", &p);

  /* leitura dos pontos */
  for (i = 0; i < n; i++)
    {
      printf("Digite o %do. ponto: ");
      for (j = 0; j < d; j++)
        scanf("%f", &A[i][j]);
    }

  /* copie origem e primeiro ponto */
  for (j = 0; j < d; j++)
    origem[j] = A[n-1][j];

  /* ponto 0 e' o de menor distancia ate que se prove o contrario*/
  diferenca(A[0],origem,diff,d);
  mindist  = norma(diff,d,p);
  imindist = 0;
```

```
  for (i = 1; i < n-1; i++)
    {
      /* calcule a distancia entre x e a origem */
      diferenca(A[i],origem,diff,d);
      dist = norma(diff,d,p);

      /* verifique se o ponto i esta mais proximo da origem */
      if (dist < mindist)
        {
          mindist  = dist;
          imindist = i;
        }
    }

   printf("O ponto mais proximo do ponto origem e' = (");
   for (j = 0; j < d-1; j++)
     {
       printf("%f,", A[imindist][j]);
     }
   printf("%f)\n cuja distancia e' = %f\n", A[imindist][d-1], mindist);

   return 0;
}
```

QUESTÃO 3 (vale 3,5 pontos) Após o grande sucesso do jogo Lig-k no Brasil todo, a fabricante de EPs Bixos da Poli resolveu desenvolver um jogo semelhante para iniciantes. A grande parte das queixas dos consumidores do Lig-k era que o jogo era complexo demais para seus filhos de menos de 5 anos de idade. Logo, surgiu a idéia de se desenvolver um jogo mais simples, em que somente as linhas e colunas do tabuleiro definem a partida. Em outras palavras, as diagonais do tabuleiro não são levados em conta.

Sua missão: desenvolver "parte" do jogo Lig-k para *beginners* durante as duas horas de prova.

Para este exercício, considere que o tabuleiro representa um jogo com no máximo um vencedor.

Além disso, para manter compatibilidade com a versão anterior, o tabuleiro deve ter uma moldura com as mesmas especificações do EP4. Para isso, considere a definição da constante

```
#define MOLDURA 'm'
```

utilizada para preenchimento da moldura.

PÁGINA DEIXADA EM BRANCO POR QUESTÕES DE FORMATO

ITEM 3(a) (vale 1,5 ponto) Escreva uma função de protótipo

```
int verifica(int n, char v[MAX], int k) ;
```

que recebe um inteiro `n`, um vetor de caracteres `v` que pode conter os caracteres 'X' ou 'O' ou um espaco vazio (' '), nas posições de índices de 1 até `n` (considere que o caractere MOLDURA está nas posições 0 e $n+1$) , e um inteiro `k` que representa quantos 'X' ou 'O' devem aparecer de forma consecutiva para que haja vencedor. A funcão deve verificar se houve ganhador ou não, ou seja, se há uma sequência contígua de `k` caracteres 'X' ou `k` caracteres 'O'. Se houver ganhador, devolver 0 caso o 'X' ganhe, 1 caso o 'O' ganhe, ou -1 caso não haja vencedor.

```
/*
 * Solucao 1:
 */
int verifica(int n, char v[MAX], int k)
{
  int i;
  int contX; /* contador de 'X's consecutivos no vetor */
  int contO; /* contador de 'O's consecutivos no vetor */
  int resp;  /* valor a ser retornado */

  resp = -1;
  contX = contO = 0;
  for (i = 1; i <= n; i++)
    {
      if (v[i] == 'X')
        {
          contX = contX + 1;
          contO = 0;
        }
      else if (v[i] == 'O')
        {
          contO = contO + 1;
          contX = 0;
        }
      else  /* v[i] == ' ' */
        {
          contX = contX = 0;
        }

      if (contX == k) resp = 0;
      else if (contO == k) resp = 1;
    }

  return resp;
}
```

```
/*
 * Solucao 2: usa vários return
 */
int verifica(int n, char v[MAX], int k)
{
  int i;
  int contX; /* contador de 'X's consecutivos no vetor */
  int cont0; /* contador de '0's consecutivos no vetor */

  contX = cont0 = 0;
  for (i = 1; i <= n; i++)
    {
      if (v[i] == 'X')
        {
          contX = contX + 1;
          cont0 = 0;
        }
      else if (v[i] == '0')
        {
          cont0 = cont0 + 1;
          contX = 0;
        }
      else  /* v[i] == ' ' */
        {
          contX = contX = 0;
        }

      if (contX == k) return 0;
      else if (cont0 == k) return 1;
    }

  return -1;
}
```

ITEM 3(b) (vale 2,0 pontos)

Escreva um programa em C que, usando a função `LeiaDados`, lê dois inteiros `nlinhas` e `ncolunas`; uma matriz de caracteres `tabuleiro` (configuracão de jogo do Lig-k) de dimensão `nlinhas` × `ncolunas`; e um inteiro `k` que representa quantos discos devem estar ligados para que haja vencedor.

Seu programa deve verificar se na configuração lida há um vencedor. Caso haja, deve imprimir quem é o vencedor (o 'X' ou o 'O') e em qual linha ou coluna encontra-se a sequência consecutiva de `k` discos; caso não haja, deve imprimir que não há uma sequência de `k` discos consecutivos.

Para a leitura dos dados, você **deve** usar a funcão com protótipo

```
void LeiaDados(char tabuleiro[MAX][MAX], int *k, int *nlinhas, int *ncolunas) ;
```

que coloca na matriz de caracteres `tabuleiro` a configuracão do jogo, incluindo também no tabuleiro uma moldura de caracteres MOLDURA; no inteiro `*k` o número de discos consecutivos para que haja vencedor; e nos inteiros `*nlinhas` e `*ncolunas` o número de linhas e colunas (**não** incluindo nesta contagem as linhas e colunas referentes à moldura) da matriz `tabuleiro`, respectivamente. Você **NÃO** precisa escrever esta função: considere que ela já está feita.

Além disso, seu programa também deve utilizar **OBRIGATORIAMENTE** a funcão do item (a) mesmo que você não a tenha feita.

Para este exercício, considere que o tabuleiro representa um jogo com no máximo um vencedor . Considere ainda que, no caso de haver uma linha e uma coluna que fazem um jogador vencedor, o seu programa pode imprimir ou a linha ou a coluna correspondente.

```
/*
 * Solucao 1:
 */
int main()
{
  char tabuleiro[MAX][MAX];
  char v[MAX];   /* para copia de uma linha ou coluna do tabuleiro */
  int nlinhas;
  int ncolunas;
  int k;
  int vencedor;

  LeiaDados(tabuleiro, &k, &nlinhas, &ncolunas);

  vencedor = linha = coluna = -1;

  /* verifica venceu na linha */
  for (i = 1; i <= nlinhas; i++)
   {
     for (j = 1; j <= ncolunas; j++)
       v[j] = tabuleiro[i][j];

     vencedor = verifica(ncoluna,v,k);
     if (vencedor != -1)
       linha = i;
   }

  /* verifica venceu na linha */
  for (j = 1; j <= ncolunas; i++)
   {
     for (i = 1; i <= nlinhas; i++)
       v[i] = tabuleiro[i][j];

     vencedor = verifica(nlinhas,v,k);
     if (vencedor != -1)
       linha = i;
   }

   if (vencedor == 0)
     printf("Jogador X venceu na ");
   else  if (vencedor == 1)
     printf("Jogador O venceu na ");
   else
     printf("Nao ha uma sequencia de k discos consecutivos\n");

   if (linha > 0)  printf("linha  %d\n", linha);
   if (coluna > 0) printf("coluna %d\n", coluna);

   return 0;
}
```

```
/*
 * Solucao 2:
 */
int main()
{
  char tabuleiro[MAX][MAX];
  int nlinhas;
  int ncolunas;
  int k;
  int vencedor;

  LeiaDados(tabuleiro, &k, &nlinhas, &ncolunas);

  vencedor = linha = coluna = -1;

  /* verifica venceu na linha */
  for (i = 1; i <= nlinhas; i++)
   {
     vencedor = verifica(ncoluna,tabuleiro[i],k);
     if (vencedor != -1)
       linha = i;
   }

  /* verifica venceu na linha */
  for (j = 1; j <= ncolunas; i++)
   {
     for (i = 1; i <= nlinhas; i++)
       v[i] = tabuleiro[i][j];

     vencedor = verifica(nlinhas,v,k);
     if (vencedor != -1)
       linha = i;
   }

   if (vencedor == 0)
     printf("Jogador X venceu na ");
   else  if (vencedor == 1)
     printf("Jogador O venceu na ");
   else
     printf("Nao ha uma sequencia de k discos consecutivos\n");

   if (linha > 0)  printf("linha  %d\n", linha);
   if (coluna > 0) printf("coluna %d\n", coluna);

   return 0;
}
```