MAC2166 – Introdução à Computação para Engenharia

Escola Politécnica

Terceira Prova – 20 de junho de 2011

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 permitido o uso de folhas avulsas para rascunho.
6. Não é permitido o uso de calculadoras.
7. 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)	2,0	
3(a)	1,0	
3(b)	1,5	
3(c)	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>

int  f0(int nusp, int v[8]);

void f1(int n, int v[8]);

int  f2(int n, int v[8], int m[2][8]);

void f3(int *a, int i, int j, int m[2][8]);

int main() {
  int nusp, n, c, i, j, k, v[8], m[2][8];

  printf("Digite o seu no. USP: ");
  scanf("%d", &nusp); /* use o SEU no. USP */
  printf("0: nusp = %d\n", nusp);

  n = f0(nusp, v);
  printf("1: n = %d  ", n);
  f1(8, v);

  c = f2(8, v, m);
  printf("2: c = %d\n", c);
  printf("3: ");
  f1(4, m[0]);
  printf("4: ");
  f1(4, m[1]);

  i = 1;
  j = 2;
  k = 10;
  f3(&k, i, j, m);
  printf("5: i = %d  j = %d  k = %d  ", i, j, k);
  f1(4, m[i]);

  m[0][0] = 1;
  m[0][1] = 3;
  m[0][2] = 12;
  m[0][3] = 25;
  printf("6: ");
  f1(4, m[0]);
  f3(&m[0][2], m[0][0], m[0][1], m);
  printf("7: ");
  f1(4, m[0]);
  printf("8: ");
  f1(4, m[1]);

  return 0;
}
```

```
int f0(int nusp, int v[8]) {
  int n;

  for (n = 0; n < 8; n++) {
    v[n] = nusp % 10;
    nusp = nusp / 10;
  }
  return n;
}
```

```
void f1(int n, int v[8]) {
  int i;

  printf("[%d", v[0]);
  for (i = 1; i < n; i++)
    printf(", %d", v[i]);
  printf("]\n");
}
```

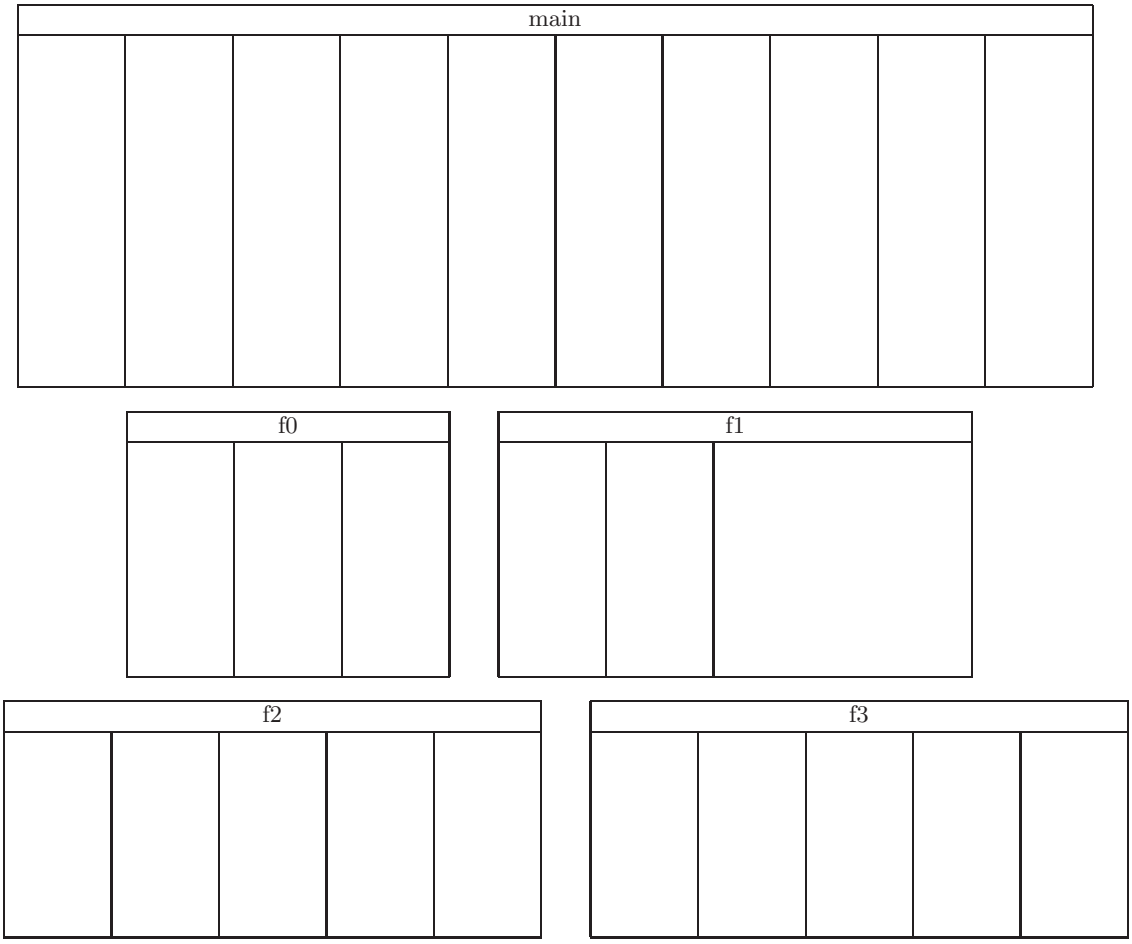
```
int f2(int n, int v[8], int m[2][8]) {
  int i, t;

  t = n/2;
  for (i = 0; i < n; i++)
    m[i/t][i%t] = v[i];
  return t;
}
```

```
void f3(int *a, int i, int j, int m[2][8]) {
  int aux;

  aux = *a;
  *a = m[i][j];
  m[i][j] = aux;
}
```

Você pode usar a tabela abaixo como bem entender. Cada turma está habituada a simular de maneira diferente, fazendo tabelas com "caras" diferentes da abaixo.



Saída do programa

Questão 2 (vale 3 pontos)

Uma sequência de n números é *mágica* se o i-ésimo número da sequência, denotado por x_i, é o número de vezes que i aparece na sequência, para $i = 0, \ldots, n-1$.

Exemplo 1: A sequência com 5 números $\mathbf{2}, \mathbf{1}, \mathbf{2}, \mathbf{0}, \mathbf{0}$ é mágica, pois
0 aparece **2** vezes na sequência e $x_0 = \mathbf{2}$,
1 aparece **1** vez na sequência e $x_1 = \mathbf{1}$,
2 aparece **2** vezes na sequência e $x_2 = \mathbf{2}$,
3 aparece **0** vezes na sequência e $x_3 = \mathbf{0}$, e
4 aparece **0** vezes na sequência e $x_4 = \mathbf{0}$.

Exemplo 2: A sequência com 7 números $3, 2, 1, \mathbf{0}, 1, 0, 0$ não é mágica,
pois o 3 aparece 1 vez na sequência mas $x_3 = \mathbf{0}$.

Exemplo 3: A sequência com 7 números $\mathbf{3}, \mathbf{2}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}$ é mágica, pois
0 aparece **3** vezes na sequência e $x_0 = \mathbf{3}$,
1 aparece **2** vezes na sequência e $x_1 = \mathbf{2}$,
2 aparece **1** vez na sequência e $x_2 = \mathbf{1}$,
3 aparece **1** vez na sequência e $x_3 = \mathbf{1}$,
4 aparece **0** vezes na sequência e $x_4 = \mathbf{0}$,
5 aparece **0** vezes na sequência e $x_5 = \mathbf{0}$, e
6 aparece **0** vezes na sequência e $x_6 = \mathbf{0}$.

Para esta questão suponha que está definida

```
#define MAX 100
```

---

item (2a) (vale 1 ponto) Escreva uma função de protótipo:

```
int conte (int n, int v[MAX], int i);
```

que recebe um inteiro `n`, com $0 <$ `n` $\leq$ `MAX`, um vetor `v` com `n` inteiros e um inteiro `i` ≥ 0, e retorna o número de vezes que `i` aparece no vetor `v`.

item (2b) (vale 2 pontos) Escreva um programa que lê um inteiro positivo `n` e uma sequência de `n` inteiros e imprime "`A sequencia eh magica!`" se a sequência dada for mágica, e imprime "`A sequencia nao eh magica...`", em caso contrário. O seu programa deve usar **obrigatoriamente** a função `conte`. Você pode usar essa função **mesmo sem a ter escrito**.

Questão 3 (vale 4,5 pontos)

Considere uma rede representada exatamente como no EP4. Ou seja, a rede é formada por usuários, identificados por nomes simples como "ana", "beto", "caralegal", escritos todos em letras minúsculas. Cada usuário possui uma **lista de contatos** (LC). Como no EP, o fato de "ana" estar na LC de "beto" não necessariamente implica que "beto" está na LC de "ana". Uma rede desse tipo com n usuários pode ser representada por uma matriz M de 0s e 1s, de tamanho $n \times n$, chamada de **matriz de contatos**. A linha i dessa matriz armazena a LC do usuário associado ao índice i, que denotamos U_i. Assim, temos

$$\mathrm{M}[i][j] = \begin{cases} 1, & \text{se o usuário } U_j \text{ está na LC do usuário } U_i \\ 0, & \text{caso contrário.} \end{cases} \quad (1)$$

O nome dos usuários podem ser armazenados em uma matriz `U` de caracteres, um nome em cada linha dessa matriz.

Dizemos que um usuário U_i é *famoso* se U_i está na LC de todos os outros usuários.

Suponha que as seguintes definições são dadas:

```
#define MAX 50
#define MAXNOME 16
#define MAXFILENAME 256
```

---

item (3a) (vale 1 ponto) Escreva uma função de protótipo:

```
int famoso (int n, int M[MAX][MAX], int i);
```

que recebe o número de usuários em `n`, a matriz de contatos em `M`, e um índice `i`, com $0 \leq$ `i` $<$ `n`, e retorna 1 se o usuário de índice `i` é famoso, 0 em caso contrário.

item (3b) (vale 1,5 pontos) Escreva uma função de protótipo:

```
void ordene_famosos (char U[MAX][MAXNOME], int t, int Ord[MAX]);
```

que recebe os nomes dos usuários da rede em `U`, um inteiro `t` com o número de usuários famosos na rede, e um vetor `Ord` com os índices dos `t` usuários famosos da rede. A função deve alterar `Ord` colocando os índices dos usuários famosos em ordem alfabética dos respectivos nomes. A matriz `U` não deve ser alterada. Para comparar nomes, você pode usar a função `strcmp` da biblioteca `string.h`.

item (3c) (vale 2 pontos) Para este item, utilize, **sem ter que escrever**, a função

```
int leia_rede(char nome_arquivo[MAXFILENAME], int *n, char U[MAX][MAXNOME], int M[MAX][MAX]);
```

que recebe o nome de um arquivo em `nome_arquivo`, devolve a rede (número de usuários em `*n`, lista de usuários em `U` e a matriz de contatos em `M`) que está armazenada no arquivo, e retorna 1 em caso de sucesso na leitura e 0 em caso de erro.

Escreva um programa que lê o nome de um arquivo de entrada, usa a função `leia_rede` para ler uma rede deste arquivo e, em caso de sucesso na leitura, imprime na tela a lista dos usuários famosos desta rede em ordem alfabética. O seu programa deve usar **obrigatoriamente** as funções `leia_rede`, `famoso` e `ordene_famosos`. Você pode usar essas funções **mesmo sem tê-las escrito**.