Processo de seleção PPGCC 2015/1 - 04/02/2015

Universidade Federal de Ouro Preto
Departamento de Computação - DECOM
Programa de Pós-Graduação em Ciência da Computação - PPGCC

Nome do candidato: _____

Aviso: os programas devem ser feitos em C ou C++ Valor da prova: 10 pontos

1) (1,5 pontos) Faça um programa que faça a multiplicação de duas matrizes. O tamanho das matrizes será definido pelo usuário (considere que as matrizes já estão alocadas) como no exemplo abaixo:

```
Digite o número de linhas da matriz A: 3
Digite o número de colunas da matriz A: 3
Digite o número de linhas da matriz B: 3
Digite o número de colunas da matriz B: 3
Digite o elemento da linha 0 e coluna 0 da matriz A: 5
Digite o elemento da linha 0 e coluna 1 da matriz A: 7
Digite o elemento da linha 0 e coluna 2 da matriz A: 3
Digite o elemento da linha 1 e coluna 0 da matriz A: 3
Digite o elemento da linha 1 e coluna 1 da matriz A: 4
Digite o elemento da linha 1 e coluna 2 da matriz A: 6
Digite o elemento da linha 2 e coluna 0 da matriz A: 8
Digite o elemento da linha 2 e coluna 1 da matriz A: 3
Digite o elemento da linha 2 e coluna 2 da matriz A: 2
Digite o elemento da linha 0 e coluna 0 da matriz B: 5
Digite o elemento da linha 0 e coluna 1 da matriz B: 7
Digite o elemento da linha 0 e coluna 2 da matriz B: 3
Digite o elemento da linha 1 e coluna 0 da matriz B: 3
Digite o elemento da linha 1 e coluna 1 da matriz B: 4
Digite o elemento da linha 1 e coluna 2 da matriz B: 6
Digite o elemento da linha 2 e coluna 0 da matriz B: 8
Digite o elemento da linha 2 e coluna 1 da matriz B: 3
Digite o elemento da linha 2 e coluna 2 da matriz B: 2
A matriz C = A * B é igual a:

70 72 63
75 55 45
65 74 46
```

2) (1,5 pontos) Codifique um programa que leia a nota e a frequência de um aluno e gere a resposta da situação do aluno de acordo com a tabela:

Condição	Situação
Frequência até 75%	Reprovado
Frequência entre 75% e 100%, e Nota até 3.0	Reprovado
Frequência entre 75% e 100%, e Nota de 3.0 até 7.0	Exame Especial
Frequência entre 75% e 100%, e Nota de 7.0 até 10.0	Aprovado

3) (1,5 pontos) Alguns preenchimentos válidos de árvores binárias de busca podem prejudicar severamente o desempenho das consultas realizadas na mesma.

a. explique textualmente e com um exemplo gráfico o caso de uma árvore binária de busca corretamente preenchida cuja organização deixará as consultas ineficientes.
b. explique, novamente utilizando texto e desenhos, as operações que podem ser realizadas para ajustar a árvore binária de busca de modo que o desempenho das consultas melhore.

4) (1,5 pontos) Considere uma tabela *hash* onde a função de espalhamento tem uma distribuição praticamente uniforme dos valores *hash*. Considere também que o tratamento de colisões é feito com o auxílio de uma árvore binária de busca para cada posição da tabela. Descreva e justifique textualmente, utilizando notação assintótica, e com desenhos qual seria o custo da operação de pesquisa no melhor caso e pior caso.

5) (2 pontos) Suponha a existência das funções abaixo:

```
void troca(int &a, int &b){
    int aux;
    aux = a;
    a = b;
    b = aux;
}

int minimo(int a[], int esq, int dir){
    int menor = esq;
    for (int i=esq+1; i<=dir; i++){
        if (a[i] < a[menor]){
            menor = i;
        }
    }
     return menor;
}

void posiciona(int a[], int k){
     int x = a[k];
     int j = k - 1;
     while (x < a[j] && j >= 0){
          a[j+1] = a[j];
          j--;
     }
     a[j+1] = x;
}
```

a. (0,5 pontos) Utilize as funções acima e implemente uma função com o seguinte protótipo: `void selecao(int a[], int n)`
que faça uma ordenação por **seleção** em um arranjo `a[]` de tamanho `n`.

b. (0,5 pontos) Utilize as funções acima e implemente uma função com o seguinte protótipo: `void insercao(int a[], int n)`
que faça uma ordenação por **inserção** em um arranjo `a[]` de tamanho `n`.

c. (1 ponto) Considerando o número de comparações de chaves em notação assintótica, em que casos uma função é melhor que a outra?

6) (2 pontos) Caça palavras: para procurar por mensagens escondidas nos jornais, foi pedido um programa que procure por algumas palavras-chave que podem aparecer tanto na vertical quanto na horizontal.

```
// número de palavras-chave
int nkeywords = 2;
// Palavras-chave
char keywords = { "cia", "terror" };
// Número de linhas e colunas no texto
const int lines = 5;
const int columns = 80;
// Texto
char text[lines][columns] = {  "Even tough nobody expected,   ",
                               "in a clean sweep for students ",
                               " comming, Bravin announced    ",
                              "what appeared to be a new form",
                              "of interaction among people   "  };
```

Escreva uma função que tenha o seguinte protótipo:

```
void busca(int nkeywords, char *keywords, int lines, columns, char
**text)
```

onde temos, respectivamente, o numero de palavras-chave, as palavras-chave, o número de linhas, colunas e conteúdo da matriz de texto.

Sua função deve procurar pelas palavras-chave, iniciando em qualquer posição da matriz e, escritas tanto na vertical quanto na horizontal. Ao encontrar uma palavra-chave sua função deve imprimir:

"Encontrado (linha,coluna) {V,H} palavra"

Como por exemplo:

```
Encontrado (1,5) V  cia
```

onde (1,5) indicam a linha e a coluna, V indica que o texto foi encontrado na vertical seguido pelo texto encontrado.