Universidade Federal de Ouro Preto – UFOP

Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: BCC201 – Introdução a Programação
Professores: Puca Huachi, Fernanda Souza e Rafael Gomes

Prova 3 - 2021-2

Questão A	Questão B	Questão C	Questão D	**Nota Total**

Nome: _____ Matrícula: _____ Turma: _____

Questão 01 (20% da nota da prova)

Leia o código a seguir com **muita atenção**. Em seguida, aponte **quatro** erros/problemas com o código e mostre como resolvê-los (apresente de forma clara o código fonte da resolução):

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int mat;
    float *notas;
    float media;
} Aluno;

// função que retorna o índice do aluno que tem a matricula passada por parâmetro
// ou -1 caso o aluno não exista no vetor passado por parâmetro
int encontraAluno(Aluno *dados, int n, int matricula) {
    for (int i = 0; i < n; i++) {
        if (dados[i]->mat == matricula)
            return dados[i];
    }
    return -1;
}

int main()
{
    int n, m;
    printf("Digite o nro de alunos (n) e de notas (m): ");
    scanf("%d %d", &n, &m);

    Aluno *dados = malloc(n);

    // lendo os dados dos alunos
    for (int i = 0; i < n; i++) {
        printf("Digite o nro de matricula do aluno %d: ", i+1);
        scanf("%d", &dados[i].mat);
        dados[i].notas = malloc(m * sizeof(int));
```

```c
        dados[i].media = 0;
        printf("Digite as %d notas do aluno %d: ", m, i+1);
        for (int j = 0; j < m; j++) {
            scanf("%f", &dados[i].notas[j]);
            dados[i].media += dados[i].notas[j] / m;
        }
    }

    // calculando e imprimindo a media total
    float mediaTotal = 0;
    for (int i = 0; i < n; i++)
        mediaTotal += dados[i].media / n;
    printf("Media geral: %f\n", mediaTotal);

    // solicitando nro de matricula e imprimindo dados do aluno
    int mat;
    int indice;
    do {
        printf("Digite um nro de matricula válido: ");
        scanf("%d", &mat);
        indice = encontraAluno(dados, n, mat);
    } while(indice == -1);

    printf("Media do aluno com matricula %d: %f\n", mat, dados[indice].media);

    return 0;
}
```

Questão 02 (20% da nota da prova)

Crie um programa que lê um número n e o **primeiro nome** (até 50 caracteres), **nota** e **frequência** de n alunos para, em seguida, armazenar estes dados em dois arquivos: `alunos.txt` (arquivo de texto) e `alunos.dat` (arquivo binário).

Você deve definir um tipo `Aluno` para representar um aluno e criar duas funções para salvar os dados: uma função `salvarTxt` para gerar o arquivo `aluno.txt` e uma função `salvarDat` para gerar o arquivo `alunos.dat`. Fique a vontade para escolher os formatos dos arquivos gerados, desde que toda a informação seja devidamente armazenada.

Importante: utilize alocação dinâmica para ler os dados do usuário.

Questão 03 (30% da nota da prova)

Implemente uma função em C que lê um arquivo de texto contendo no máximo 100 linhas com até 100 caracteres cada, substitui os espaços pelo caractere '_', e em seguida cria um arquivo com as linhas (após a substituição) em ordem inversa.

A função deve ter a seguinte assinatura:

```c
// entrada e saida são as strings que indicam as localizações dos arquivos
void processaArquivoTexto(char entrada[], char saida[]);
```

Dica: use a função `fgets` da biblioteca `<stdio.h>` para ler uma linha completa do arquivo. Não se esqueça que esta função (`fgets`) inclui o '\n' ao ler a linha.

Questão 04 (30% da nota da prova)

No jogo *CrazyTab* cada posição do tabuleiro contém um número inteiro e é representada por suas coordenadas x e y. O tabuleiro pode ser representado por uma matriz, tal como a seguir.

x↓ y→	1	2	3	4	5	6	7	8
1	0	3	5	0	0	3	0	3
2	0	0	0	0	1	0	9	5
3	3	0	5	1	1	3	0	0
4	0	0	0	0	0	0	7	4
5	0	6	2	4	3	2	0	1
6	2	0	0	0	0	0	0	0

Os jogadores iniciam o jogo na posição (1,1) do tabuleiro, tendo zero pontos. Em cada rodada, eles podem se mover um certo número de casas para cima (1), para baixo (2), para a esquerda (3) ou para a direita (4). Escreva um programa que implemente o jogo para 2 jogadores, atendendo aos seguintes requisitos:

- As dimensões da matriz $M_{p \times q}$, que representam o tabuleiro, devem ser solicitadas ao usuário. Note que p é o número de linhas e q o número de colunas. Utilize alocação dinâmica.
- A matriz deve ser inicializada aleatoriamente, com cerca de 50% das células com valores entre 1 e 9 e o restante com valor zero. Dica: use a função `rand()` da biblioteca `<stdlib.h>` para definir se o valor será 0 ou não (`rand() % 2`) e, caso o valor não seja 0, utilize novamente a função `rand()` para definir o valor da célula.
- No início do programa, deve ser lido o número n de rodadas do jogo.
- Em cada rodada, o programa lê o movimento do jogador 1 e o movimento do jogador 2, sendo que para cada movimento 2 valores são lidos: a direção do movimento (1 a 4) e o número de casas a mover. Se um jogador mover-se para uma posição fora do tabuleiro ou indicar uma direção de movimento inválida, ele não acumula pontos e volta para a posição inicial (1,1).
- Cada jogador vai acumulando os pontos marcados na posição do tabuleiro em que ele parar depois de cada jogada.
- O programa deve imprimir o número de pontos de cada jogador ao final das n jogadas. Ganha o jogador que tiver o maior número de pontos.

Exemplo de execução:

```
=== CrazyTab Game ===

Digite as dimensões do tabuleiro: 6 8
Digite o número de jogadas: 3
Para cada rodada digite a direção e o número de casas a mover:

Rodada 1
   Jogador 1: 2 2
   Jogador 2: 1 2
Rodada 2
   Jogador 1: 4 2
   Jogador 2: 4 5
Rodada 3
   Jogador 1: 1 2
   Jogador 2: 2 4

Pontuação do jogador 1 = 13
Pontuação do jogador 2 = 5

=== Jogador 1 venceu ===
```