Universidade Federal de Ouro Preto - UFOP
Disciplina: BCC 201 - Introdução à Programação
Professor: Guillermo Cámara-Chávez

Aluno: _____ **No.** _____

A cola não será tolerada. Se alguém for pego colando, será reprovado com Zero. É considerado cola: olhar/copiar da prova de outro ou deixar outro aluno olhar sua prova.

3ra. Avaliação - Grupo B

1. (2pts) Você deverá implementar uma estrutura TConjunto para representar conjuntos de números inteiros. Sua estrutura deverá armazenar os elementos do conjunto e o seu tamanho n. Considere que o tamanho máximo de um conjunto é 20 elementos e use arranjos de 1 dimensão (vetores) para a sua implementação. Crie a função que permite calcular a interseção de dois conjuntos

```
typedef struct Conjunto{
    int elem[20];
    int n; // numero de elementos dentro
}TConjunto;

TConjunto Intersecao(TConjunto A, TConjunto B)
{
    TConjunto C;
    C.n = 0;
    int i, j;
    for (i = 0; i < A.n; i++)
        for (j = 0; j < B.n; j++)
            if (A.elem[i] == B.elem[j])
            {
                C.elem[C.n] = A.elem[i];
                C.n++;
                break;
            }
    return C;
}
```

2. (2pts) Crie uma função que permita alocar memória para uma matriz tridimesional. O tamanho da matriz deve ser fornecido como parâmetro de entrada. Utilize a menor quantidade possível de malloc/calloc

   Com número $lin + lin * col + 1$ calloc/malloc

```
int*** CriaMatriz3D(int lin, int col, int dim)
{
    int*** mat = NULL, i, j;
    mat = (int***) calloc (lin, sizeof(int**));
    for (i = 0; i < lin; i++){
        mat[i] = (int**) calloc (col, sizeof(int*));
        for (j = 0; j < col; j++)
            mat[i][j] = (int*) calloc (dim, sizeof(int));
        }
```

```
        return mat;
}
```

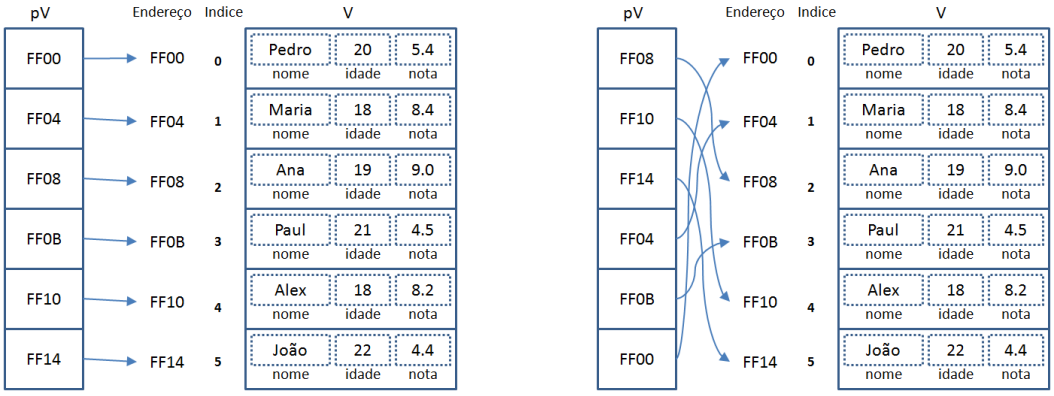
Com 3 calloc/malloc

```
int*** CriaMatriz3Dmin(int lin, int col, int dim)
{
    int*** mat = NULL, i;
    mat = (int***) calloc (lin, sizeof(int**));
    mat[0] = (int**) calloc (lin*col, sizeof(int*));
    for (i = 1; i < lin; i++)
        mat[i] = &mat[0][i*col];
    mat[0][0] = (int*) calloc (lin*col*dim, sizeof(int));
    for (i = 1; i < lin*col; i++)
        mat[0][i] = &mat[0][0][dim*i];
    return mat;
}
```

3. (2 pts) Seja a estrutura

```
typedef struct Aluno{
    char nome[100];
    int idade;
    float nota;
}TAluno;
```

Elabore um procedimento/função que receba por parâmetro um vetor de estruturas TAluno de n posições e um vetor de ponteiros a estruturas TAluno de tamanho n. Faça que cada elemento do vetor de ponteiros contenha o endereço da respectiva posição do vetor de estruturas. Logo, ordene de forma ascendente o vetor de estruturas pelo campo nome, faça a ordenação a través do vetor de ponteiros. Para ordenar os dados, modifique somente as posições do vetor de ponteiros




```
typedef struct Aluno{
    char nome[50];
    int idade;
    float nota;
}TAluno;

void OrdenaNome(TAluno* V, TAluno** pV, int n){
        int i,j;
        TAluno *tmp;
        for (i = 0; i < n; i++)
                pV[i] = &V[i];
        for (i = 0; i < n; i++)
                for (j = 0; j < n-1; j++){
                        if (strcmp(pV[j]->nome, pV[j+1]->nome) > 0)
                        {
```

```
                                    tmp = pV[j];
                                    pV[j] = pV[j+1];
                                    pV[j+1] = tmp;
                                }
                            }
}
```

4. (2 pts) Um arquivo do tipo texto, chamado "numeros.txt" contém uma matriz de $m \times n$ números reais. Escreva um programa que leia estes números e salve os mesmo em uma matriz dinâmica. Para alocar memória para a matriz é necessário contar o número de linhas e de colunas.

```
FILE* pfile = fopen("numeros.txt", "r");
int contaespaco = 0, i, contaespacoTotal = 0, cont;
char car;
int **M = NULL, lin = 0, col, j;
while(!feof(pfile)){
    cont = fscanf(pfile, "%c", &car);
    if ( (car == '\n' || cont == 0)) // fim de linha ou EOF
    {
        // conta a linha, se ela tem mais de um espaco em branco
        if (contaespaco > 0){
            lin++;
            contaespacoTotal += contaespaco;
            contaespaco = 0;
        }
    }
    else
        // conta o numero de espacos em branco
        if (car == ' ')
            contaespaco++;
}
// calcula numero de colunas
col = contaespacoTotal / lin + 1;
M = (int**) calloc(lin, sizeof(int*));
for (i = 0; i < lin; i++)
    M[i] = (int*) calloc (col, sizeof(int));
rewind(pfile);
for (i = 0; i < lin; i++)
    for (j = 0; j < col; j++)
        fscanf(pfile, "%d", &M[i][j]);
fclose(pfile);
return 0;
}
```

5. (2pts) Faça uma função **MAX** que recebe como entrada um inteiro n, uma matriz de *float* $A_{n \times n}$ (utilize ponteiros) e devolve e um float k e dois vetores de inteiros: l e c. k é o maior elemento de A e é igual a $A[l_i][c_i]$. Se o elemento máximo ocorrer mais de uma vez, retornar todas as posições onde se encontra o máximo elemento.

```
int MAX(int** M, int tam, int* k, int *l, int* c)
{
    // k é o endereço de um inteiro
    // l e c são 2 vetores
    int i, j, cont;
    // encontra o maoir
    *k = M[0][0];
    for (i = 0; i < tam; i++)
        for (j = 0; j < tam; j++)
            if (M[i][j] > *k)
                *k = M[i][j];
```

```
    // se o maior estiver repetido, encontra todas as posicoes
    cont = 0;
    for (i = 0; i < tam; i++)
        for (j = 0; j < tam; j++)
            if (M[i][j] == *k){
                l[cont] = i;
                c[cont++] = j;
            }
    // retorno numero de vezes que o maior esta na matriz
    return cont;
}
```