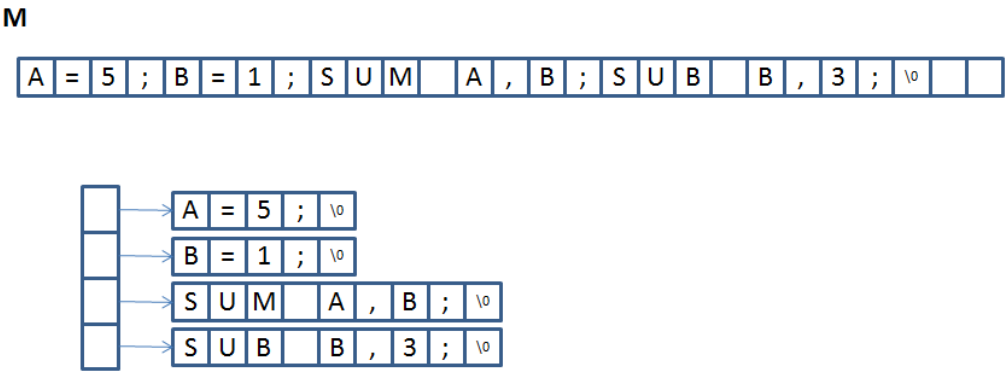
Universidade Federal de Ouro Preto - UFOP
Disciplina: BCC 201 - Introdução à Programação
Professor: Guillermo Cámara-Chávez

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 A

1. (2 pts) Seja M um vetor unidimensional de caracteres que representa a memória de um computador. Nela encontram-se as diferentes instruções de um algoritmos, casa instrução finaliza com ';'. Crie uma função que permita retornar um vetor de *strings*, onde casa posição do vetor contem um única instrução.



```
char** Intrucoes(char* M, int *n)
{
    // ENTRADA
    // M: memoria
    // n: representa o numero total de instrucoes, sera
    //    contado dentro desta funcao
    // SAIDA
    // V: vetor com as intrucoes
    int i, j = 0, tam = strlen(M), k;
    char **V, tmp[50];
    *n = 0;
    for (i = 0; i < tam; i++)
        if (M[i] == ';') (*n)++;
    V = (char**) calloc(*n, sizeof(char*));
    for (i = 0; i < *n; i++)
    {
        k = 0;
        while(M[j] != ';')
            tmp[k++] = M[j++];
        j++;
        tmp[k++] = ';';
        tmp[k] = '\0';
        V[i] = (char*) calloc(strlen(tmp)+1, sizeof(char));
        strcpy(V[i], tmp);
    }
    return V;
}
```

2. (2 pts) Foi escrito um arquivo binário com os dados de n pessoas. O arquivo foi gerado a partir dos dados definidos na estrutura *Pessoa* com os campos: *nome* e *idade*. Ordene alfabeticamente pelo campo nome os dados escritos no arquivo. Faça a ordenação diretamente no arquivo. DICA: use a função `fseek()` junto com parâmetro `SEEK_CUR` para se movimentar dentro do arquivo.

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

void Ordena(char* nomeArquivo)
{
    FILE* pf = fopen(nomeArquivo, "r+b");
    int i, j, n = 0;
    TAluno tmpA, tmpB;
    while(fread(&tmpA, sizeof(TAluno), 1, pf) != 0)
        n++;
    rewind(pf);
    for (i = 0; i < n; i++)
    {
        for (j = 1; j < n; j++)
        {
            fread(&tmpA, sizeof(TAluno), 1, pf);
            fread(&tmpB, sizeof(TAluno), 1, pf);
            if (strcmp(tmpA.nome, tmpB.nome) > 0)
            {
                fseek(pf, -2*sizeof(TAluno), SEEK_CUR);
                fwrite(&tmpB, sizeof(TAluno), 1, pf);
                fwrite(&tmpA, sizeof(TAluno), 1, pf);
            }
            fseek(pf, -sizeof(TAluno), SEEK_CUR);
        }
        rewind(pf);
    }
    fclose(pf);
}
```

3. (2pts) Considere um arquivo de dados do tipo texto com o seguinte conteúdo (lembre que é um exemplo!):

```
3
ZE SA
8.5
10.0
ANTONIO SANTOS
7.5
8.5
SEBASTIAO OLIVEIRA
5.0
6.0
```

O arquivo acima é um exemplo. Considere então que nestes arquivos a primeira linha contém o número de alunos no arquivo. As linhas seguintes contém os seguintes dados:

- nome do aluno com no máximo 50 caracteres;
- nota da primeira prova;
- nota da segunda prova.

Escreva o procedimento que imprima os nomes de todos os alunos que têm a média das duas notas menor que 7.0

```c
void Notas(char* ArquivoNome)
{
    FILE* pf = fopen(ArquivoNome, "r");
    int n, i;
    float nota1, nota2;
    fscanf(pf, "%d", &n);
    char nome[50];
    fgets(nome, 50, pf); // pula a prox linha
    for (i = 0; i < n; i++)
    {
        fgets(nome, 50, pf);
        fscanf(pf, "%f %f", &nota1, &nota2);
        if ( (nota1 + nota2) / 2 < 7 )
            printf("%s \n", nome);
        fgets(nome, 50, pf); // pula a prox linha
    }
    fclose(pf);
}
```

ou

```c
void Notas2(char* ArquivoNome)
{
    FILE* pf = fopen(ArquivoNome, "r");
    int n, i;
    float nota1, nota2;
    char linha[50], nome[50];
    fgets(linha, 50, pf); // pula a prox linha
    n = atoi(linha);
    for (i = 0; i < n; i++)
    {
        fgets(nome, 50, pf);
        fgets(linha, 50, pf);
        nota1 = atof(linha);
        fgets(linha, 50, pf);
        nota2 = atof(linha);
        if ( (nota1 + nota2) / 2 < 7 )
            printf("%s \n", nome);
    }
    fclose(pf);
}
```

4. (1 pt) Seja o seguinte trecho de programa:

```c
int i = 3, j = 5;
int *p, *q;
p = &i;
q = &j;
```

Qual é o valor das seguinte expressões?

(a) $p = \&i$ endereço de i

(b) $3 * - * p/(*q) + 7 = 3 * -3/5 + 7$

5. (1 pt) Qual será a saída deste programa?

```c
int main(){
    int i = 7, *p = &i;
}
```

Qual é o valor das seguintes expressões: (indicar se alguma expressão é invalida)

(a) $*p + i; = 7 + 7$

(b) $\&p$; endereço de i

(c) $10 * *p; = 10 * 7$

(d) $i - p$; inválido, i é um inteiro, p é um ponteiro

(e) $p++$; aponta ao próximo inteiro, avança mais 4 Bytes

6. (2 pts) Vamos supor que um número real seja representado por uma estrutura em C, como esta:

```c
struct realtype{
    int left, right;
};
```

onde `left` e `right` representam os dígitos posicionados à esquerda e à direita do ponto decimal, respectivamente. Se `left` for um inteiro negativo, o número real representado será negativo. Escreva uma função que aceite essa estrutura e retorne o número real representado por ela.

```
double realtype2double(struct realtype num)
{
    float parte_decimal;
    parte_decimal = num.right;
    while(parte_decimal > 1)
        parte_decimal = parte_decimal / 10.0;
    if (num.left < 0)
        return (num.left - parte_decimal);
    else
        return (num.left + parte_decimal);
}
```