UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA (FEELT)

Prova 3

Métodos e Técnicas de Programação

Alunos: Amanda De Oliveira Cândido — **Matrícula:** 11621EEL023
Victor Almeida Barcelos — **Matrícula:** 11621EAU021
Wilgner Ferreira Nóbrega — **Matrícula:** 11621EAU011

Uberlândia, 11 de Dezembro de 2017.

Questão 1

MAT0 = 11621EEL023; **MAT1 =** 11621EAU021; **MAT2 =** 11621EAU011
KANO0 = 3; **KCUR0 =** 4; **KNUM0 =** 6
KANO1 = 3; **KCUR1 =** 1; **KNUM1 =** 4
KANO2 = 3; **KCUR2 =** 1; **KNUM2 =** 3

Questão 2

- O problema de coerência é que o programa não está calculando a média, só está somando os números
- O que arrisca vazar a memória é a falta da função free( ) na função em que foi usada a função malloc.

Parte do código corrigido:

```
float media_de_aleatorios(int ID) {

   int * p = (int *) malloc(N*sizeof(int));

   int i;

   float media = 0;

   for(i = 0; i < N; i++) {

      p[i] = rand()%9 + 1;

      media += p[i];

   }

   media = media/N;

   free(p);

   return media;

}
```

Questão 3:

Letra A)

- Codigo: 175

Letra B)

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define KANO0 3 // trocar por valor devido

#define KANO1 3 // trocar por valor devido

#define KANO2 3 // trocar por valor devido

#define KCUR0 4 // trocar por valor devido

#define KCUR1 1 // trocar por valor devido
```

```
#define KCUR2 1 // trocar por valor devido
#define KNUM0 6 // trocar por valor devido
#define KNUM1 4 // trocar por valor devido
#define KNUM2 3 // trocar por valor devido



 double media(double a, double b, double c) {
    return (a+b+c)/3;
}

int main() {
    int ID0 = (KANO0+KANO1+KANO2)%9 + 1,
        ID1 = (KCUR0+KCUR1+KCUR2)%9 + 1,
        ID2 = (KNUM0+KNUM1+KNUM2)%9 + 1;
    FILE * arq;
    int idA, idB, idC;
    double nA, nB, nC;
    arq = fopen("dados.dat","rb");
    if(arq == NULL) {
        fprintf(stderr,"Arquivo inexistente!\n");
        return EXIT_FAILURE;
    }
    switch(ID2) {
        case 1: idA = 13; idB = 14; idC = 64; break;
        case 2: idA = 21; idB = 42; idC = 84; break;
        case 3: idA = 23; idB = 37; idC = 46; break;
        case 4: idA = 16; idB = 55; idC = 82; break;
        case 5: idA = 9; idB = 33; idC = 76; break;
        case 6: idA = 0; idB = 39; idC = 99; break;
        case 7: idA = 10; idB = 86; idC = 92; break;
        case 8: idA = 17; idB = 61; idC = 92; break;
        case 9: idA = 11; idB = 24; idC = 77; break;
        case 10: idA = 5; idB = 53; idC = 65; break;
```

```
        default: idA = idB = idC = 0;
    }
// A mudança começa aqui
    fseek(arq, idA*sizeof(double), SEEK_SET);
    fread(&nA, sizeof(double),1,arq);
    fseek(arq, idB*sizeof(double), SEEK_SET);
    fread(&nB, sizeof(double),1,arq);
    fseek(arq, idC*sizeof(double), SEEK_SET);
    fread(&nC, sizeof(double),1,arq);
// Até aqui

    fclose(arq);
    printf("Matricula: %d%d%d\n",ID0,ID1,ID2);
    printf("Media [%lf %lf %lf] = %lf\n",nA,nB,nC,media(nA,nB,nC));
    return EXIT_SUCCESS;
}
```

Questão 4:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define KANO0 3 // trocar por valor devido
#define KANO1 3 // trocar por valor devido
#define KANO2 3 // trocar por valor devido
#define KCUR0 4 // trocar por valor devido
#define KCUR1 1 // trocar por valor devido
#define KCUR2 1 // trocar por valor devido
#define KNUM0 6 // trocar por valor devido
#define KNUM1 4 // trocar por valor devido
#define KNUM2 3 // trocar por valor devido
```

```
typedef
    struct Aluno {
        char nome[256];
        int matricula;
        unsigned int idade;
    }
Aluno;

void mostrar(Aluno aluno) {
    printf("> %s: MAT %03d\n: %u anos;\n", aluno.nome, aluno.matricula,
        aluno.idade);
}

void gravar(Aluno aluno) {
    FILE * arq;
    arq = fopen("registro.txt","a"); //
    fwrite(&(aluno.nome),256,1,arq);
    fwrite(&(aluno.matricula),sizeof(int),1,arq);
    fwrite(&(aluno.idade),sizeof(unsigned int),1,arq);
    fclose(arq);
}

int ler(FILE * arq, Aluno * paluno, unsigned int id) {
    fseek(arq,id*sizeof(Aluno),SEEK_SET);
    int ok = fread(&(paluno->nome),256,1,arq);
    fread(&(paluno->matricula),sizeof(unsigned int),1,arq); //
    fread(&(paluno->idade),sizeof(int),1,arq); //
    return ok;
}

void inicia() {
```

```
    remove("registro.txt");
    Aluno aluno;
    strncpy(aluno.nome,"Oswald",256);
    aluno.matricula = rand()%999 + 1;
    aluno.idade = rand()%11 + 17;
    gravar(aluno);

    strncpy(aluno.nome,"Amanda", 256);
    aluno.matricula = 23;
    aluno.idade = 19;
    gravar(aluno);

    strncpy(aluno.nome,"Victor", 256);
    aluno.matricula = 21; // Quando colocamos o zero, a questão da erro
    aluno.idade = 19;
    gravar(aluno); // Estava faltando essa função gravar
    strncpy(aluno.nome,"Wilgner", 256);
    aluno.matricula = 11;
    aluno.idade = 19;
    gravar(aluno);
    strncpy(aluno.nome,"Silvia", 256);
    aluno.matricula = rand()%999 + 1;
    aluno.idade = rand()%15 + 17;
    gravar(aluno);
    strncpy(aluno.nome,"Mickey", 256);
    aluno.matricula = rand()%999 + 1;
    aluno.idade = rand()%9 + 17;
    gravar(aluno);
}

int main() {
```

```
    int ID0 = (KANO0+KANO1+KANO2)%9 + 1,
        ID1 = (KCUR0+KCUR1+KCUR2)%9 + 1,
        ID2 = (KNUM0+KNUM1+KNUM2)%9 + 1;
    srand(ID0*100+ID1*10+ID2);
    Aluno aluno;
    FILE * arq;
    unsigned int i;
    inicia();
    arq = fopen("registro.txt","r");
    i = 0;
    while(!feof(arq)) {
        if(ler(arq, &aluno, i))
            mostrar(aluno);
        i++;
    }
    fclose(arq);
    return EXIT_SUCCESS;
}
```