Prova de MTP
Alunos:
Barbara Cristina 11711EBI021
Felipe Porto 11621EAU020
Gabriela Porto 11711EBI017

Questão 1

MAT0 11711EBI021	**MAT1** 11621EAU020	**MAT2** 11711EBI017
KANO0 3	**KANO1** 3	**KANO2** 3
KACUR0 2	**KACUR1** 1	**KACUR2** 2
KNUM0 4	**KNUM1** 3	**KNUM2** 9

Questão 2

a) A força gravitacional entre as esferas azul e vermelha, distantes 11.752659 metros uma da outra, eh de 13.038190 Newtons.
b) O objetivo do código é calcular a força gravitacional entre duas esferas. Primeiramente, foi definido uma struct que guardará posições nos três eixos cartesianos e as massas das esferas, dados que serão dados pelo usuário. Logo após isso há a função distância, a partir dos dados enviados dentro do main no "print", a distância entre os centros das esferas nos eixos x,y,z, simulando um cálculo da norma entre pontos. Por seguinte, também dentro do "print", o main chama a função que calcula a força gravitacional, considerando que G é definido como variável global e da distância que é dada pelo função da distância ao ser chamada. Por fim, as funções retornam seus resultados nos locais onde foram chamadas.

Questão 3

a) instancia: zy 1.700000
instancia: yx 1.700000
novainsta: yx -8.300000

b) O tamanho da estrutura é de 12 bytes, sendo 8 do double e 4 do char.
c) O valor da instância (codigo e preço) é modificado por meio de operações até entrar na função criaInstancia, que recebe o ponteiro codigo por referência e modifica o valor do endereço de memória do código. O preço apesar de ser modificado no escopo da função não tem seu valor na variável original modificado, visto que foi passado por valor.
d) Os membros de instancia.codigo e novainsta.codigo sao iguais, pois são endereços de memória, que não se alteram durante a execução. Já o instacia.preco e o novainsta.preco são diferentes, pois o primeiro é passado para a função como um valor que se altera ao percorrê-la, gerando o novainsta.preco, outro valor.

Questão 4

a) *[2]* 11711EBI017 [1] 11621EAU020 [0] 11711EBI021;

b) *correção do código*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N = 3

typedef
struct stTexto {
   int id;
   char mat[12];
} Texto;

void imprime(Texto * dado, int qtde) {
   if(qtde > 0) {
   printf(": (%d) %s ", dado->id, dado->mat);
   imprime(dado+1,qtde-1);
   }
   else
      printf(":\n");
}
int main() {
   Texto grupo[N];
   int i;
   for(i = 0; i < N; i++)
      grupo[i].id = i;
   strcpy(grupo[0].mat,"11711EBI021");
   strcpy(grupo[1].mat,"11621EAU020");
   strcpy(grupo[2].mat,"11711EBI017");
   imprime(grupo, N);
   return 0;
}
```

c) É um erro do uso da recursividade, pois os resultados são mostrados na ordem decrescente, quando o necessário era mostrá los na ordem crescente.

Questão 5

a) Programa com as devidas substituições:

```
#include <stdio.h>
#include <stdlib.h>
int fib(int indice) {
static int memoria[36] = {0,1};
int condicao = (memoria[indice] || !indice);
int resposta = (condicao)? memoria[indice] :
fib(indice-1) + fib(indice-2);
return resposta;
}
int main() {
printf("Fibonacci(%d) = %d\n", 4-1, fib(4-1));
printf("Fibonacci(%d) = %d\n", 3-1, fib(3-1));
printf("Fibonacci(%d) = %d\n", 9-1, fib(9-1));
```

```
printf("Numero secreto = %d", 3*fib(4-1) +
3*fib(9-1) + 3*fib(3-1) -
2*1*2);
return 0;
}
```

Resultado:
Fibonacci(3) = 2
Fibonacci(2) = 1
Fibonacci(8) = 21
Número secreto = 68

b) A recursão de Fibonacci foi definida de modo que o usuário só pode digitar um número menor que 36 para que o programa rode sem erro, visto que a memória estática foi definida com 36 elementos. A função obedece a uma condição, logo no início é definido um vetor com os valores de Fib(0) e Fib(1), então a função checa o argumento recebido para avaliar se é 0 ou 1, se for, a resposta retorna o próprio valor contido na posição índice do vetor memoria, se diferente, é calculado os dois fib() anteriores, que sao somados, para que seja retornado o valor do número de Fibonacci de índice "n". Além disso,a recursão de Fibonacci foi definida de modo que o usuário só pode digitar um número menor que 36 para que o programa rode sem erro e a função só funciona com argumentos do tipo inteiro.

Questão 6

```
#include <stdio.h>

int trib(int n)
{
   int tribn;
   if(n<=2)
          tribn = n;
   else
         tribn= trib(n-1)+2*trib(n-2)+3*trib(n-3);
      return tribn
}
int main ()
{
   int a =0;
   for(int i=0;i<15;i++)
   {
      a = i;
     printf("%d",trib(num));
   }
return 0;
}
```

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

int media(int n, float *numero)
{
	int i,t=0;
	float m =0;
	for(i=0;i<n;i++)
	{
		t = t + numero[i];
	}
	m = t/n;
	printf("\nA media e igual a %f\n", m);
	return m;
}
int desvio(int m,float *numero, int n)
{
	int i;
	float d=0;
	for(i=0;i<n;i++)
	{
		d = d + pow((numero[i] - m),2);
	}
	d = sqrt(d/(n-1));
	printf("\nO desvio padrao e %f\n", d);
	return d;
}
int main ()
{
	float * numeros;
   int i,n,x;
   float v=0;
   printf("\nDigite quantos numeros voce deseja: ");
   scanf("%d", &n);
   numeros = (float*)calloc(n,sizeof(float));
   printf("\nDigite os numeros que voce deseja: ");
   for (i=0;i<n;i++)
   {
		scanf("%f", &numeros[i]);
   }
   v = media(n,numeros);
   desvio(v,numeros,n);

   return 0;
}
```

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

void preenche(float * elemento)
{
  scanf("%f", elemento);
}

void imprime(float **matriz, int N, int M)
{
  int i, j;
  printf("matriz %dx%d:\n", N, M);
  for(i = 0; i < N; i++)
    for(j = 0; j < M; j++)
      printf("%g%c", matriz[i][j], (j == M-1)? '\n':'\t');
}
 void transposta(float **matriz, int N, int M)
{
            int i,j;
            float  matrizt[M][N];
            for (i=0;i<M;i++)
                        for (j=0;j<N;j++)
                                    printf("%g%c", matriz[j][i], (j == M-1)? '\n':'\t');

}

int main()
{
  int i, j;
  int N, M;
  float **matriz;
  printf("Entre com a ordem da matriz, no formato 'NxM': ");
  scanf("%d %d", &N, &M);
  matriz = calloc(N,sizeof(float*));
  for(i = 0; i < N; i++) {
            matriz[i] = calloc(M,sizeof(float));
    for(j = 0; j < M; j++) {
      printf("Elemento (%d,%d): ", i, j);
      preenche(&matriz[i][j]);
    }
  }
  imprime(matriz, N, M);
  transposta(matriz,N,M);
  return 0;
}
```