Avaliação Final – AEDS I

Nome: Davi Ventura Cardoso Perdigão

Conforme instruções, todos os programas foram testados anteriormente no Dev C++, nesse anexo irei copiá-los exatamente como estavam no compilador.

1)

```
#include <stdio.h>

#include <stdlib.h>

#include <locale.h>


//Davi Perdigão - Questão 1 Avaliação Final Adriano


void inverter(int numprincipais[10]);

int main(void)

{

	setlocale(LC_ALL, "Portuguese");

  int numprincipais[10], i;


  for(i=0; i<10; i++)

	{

   printf("Digite o valor do %dº elemento: ",i+1);

   scanf("%d", &numprincipais[i]);

  }

	inverter(numprincipais);


	printf("\nElementos invertidos:\n\n");

	for(i=0; i<10; i++)

	{

   printf("\t%d ",numprincipais[i]);

  }
```

```
            printf("\n\n");


    system("pause");
    return 0;
}

void inverter(int numprincipais[10])
{
    for(int i=0; i<10; i++)
            {
      int aux=numprincipais[i];
      numprincipais[i]=numprincipais[i+1];
      numprincipais[i+1]=aux;
      i++;
    }
}
```

2)

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

//Davi Perdigão - Questão 2 Avaliação Final Adriano

void multiplicar(float m[4][4], float *pA);
int main(void)
{
            setlocale(LC_ALL, "Portuguese");
            int i,j;
            float m[4][4],A;
```

```
	for(i=0;i<4;i++)
	{
		for(j=0;j<4;j++)
		{
		  printf("Informe o %dº valor da %dº linha: ",j+1,i+1);
		  scanf("%f",&m[i][j]);
		}
		printf("\n");
	}
	printf("Informe um número para multiplicar todos os valores da matriz: ");
	scanf("%f",&A);
	multiplicar(m,&A);

	system("pause");
	return 0;
}

void multiplicar(float m[4][4], float *pA)
{
	int i,j;
	for(i=0;i<4;i++)
	{
		for(j=0;j<4;j++)
		{
		  m[i][j]=m[i][j]* *pA;
		}
	}
	printf("\n\nProduto da matriz multiplicado por %.2f\n\n",*pA);
	for(i=0;i<4;i++)
	{
		for(j=0;j<4;j++)
```

```
                    {
                      printf("\t%.2f\t",m[i][j]);
                    }
                    printf("\n");
            }
            printf("\n\n\n");
}
```

3)

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

//Davi Perdigão - Questão 3 Avaliação Final Adriano

int R(int n)
{
  double r;
  if(n == 0) return 0;
  return r = pow(n, 2) + R(n-1);
}

int main()
{
   int n;

   do
          {
     printf("\nDigite um valor inteiro para N: ");
     scanf("%d", &n);
```

```
    }
        while(n < 0);

    printf("\nResultado: %d", R(n));

    printf("\n\n");
    system("pause");
    return 0;
}
```

4)

A) A **função recursiva** é como um processo de repetição de uma rotina. No geral, pode ser definida como uma rotina que chama a si mesma, de forma direta ou indireta. Como a rotina chama a si mesma inúmeras vezes, é preciso tomar cuidado para que **não** ocorra um **LOOP** no código, fazendo com que ele se repita eternamente. Para evitar essa ocasião, é necessário definir uma condição de terminação de forma correta pois, até mesmo a condição de terminação pode ser a causa de um LOOP.

B) Segundo **Luis Damas**, em seu livro **"Linguagem C"**, a utilização de ponteiros na linguagem C é uma das características que tornam essa linguagem tão **flexível e poderosa** ao mesmo tempo.

Os **ponteiros** são **variáveis** que armazenam o endereço de memória de outras variáveis. Eles podem apontar para qualquer tipo de variável, portanto temos ponteiros para *int, float, double,* etc. Ponteiros são muito úteis quando uma variável tem que ser acessada em diferentes etapas de um código. Assim, o código pode ter **vários** ponteiros espalhados por diversas partes do programa, apontando para a variável que desejar.

Como já descrito, eles são bastante úteis, mas ao meu ver eles são **fundamentais** principalmente nos casos: Alocação dinâmica de memória, manipulação de uma array, para retornar mais de um valor em uma função, referência para listas, pilhas, árvores e grafos.

C) Existem **inúmeras vantagens** no uso de uma **função:** permitir o reaproveitamento de um código já desenvolvido, evitar que um trecho de código seja repetido dentro de um mesmo programa, permitir a alteração de um trecho de código de uma forma mais rápida (é preciso alterar apenas dentro da função que desejar), evitar que os blocos do programa não fiquem grandes demais, tornando o código bem mais simples e facilitando a leitura do mesmo e para separar o programa em blocos que possam ser logicamente compreendidos de forma isolada.

Os **parâmetros** possibilitam que se defina sobre quais dados a função deve operar. Para **definir** os parâmetros de uma função é necessário **explicitá-los** como se estive declarando uma variável, entre os parênteses do cabeçalho da função. Caso precise declarar mais de um parâmetro, basta separá-los por vírgulas.

D) Passagem de parâmetros por valor significa ultilizar dentro de uma função uma cópia do valor de uma variável, porém não permite alterar o valor original dessa. Elas são passadas como nome das variáveis da função principal.

ex: funcao_auxiliar (variável);

Passagem de parâmetros por referência significa passar uma referência da variável seu endereço de memória sendo possível alterar o conteúdo da variável original.

ex: funcao_auxiliar (&variável);