UNIVERSIDADE CATÓLICA DE ANGOLA

Faculdade de Engenharia

Lista Geral de Exercícios de Técnicas de Programação -2019

0-Simulação de Programas

0.1-Dado o programa

```
#include  <stdio.h>
#include  <stdlib.h>
int  main( )
{
  int a,b, soma, total, termo, i;
  printf (" \n Digite um par de numeros : " );
  scanf ("%d   %d  ",&a,&b);
  printf ("\n  (%d, %d)   \n ", a, b);
  total = soma = 0;
  while ( a != 0 )
  {
     total ++;
     termo = 1;
     i = 1;
     while( i <= b)
       {
          termo *= a;
          i++;
       }
     printf (" \n  Termo = %d  \n ", termo);
     soma += termo;
     printf ( " \n Soma =  %d ", soma);
     printf ( " \n Digite um par de numeros : " );
     scanf (" %d   %d ",&a,&b);
     printf (" \n  (%d, %d)  \n ", a,b);
  }
  printf (" \n  Total de Pares = %d  \n ", total);
  return 0;
}
```

Faça a simulação passo a passo para os seguintes pares de números: 2 4; 5 2 ; 7 2 ; 0 3 ; 3 3

0.2- Dado o segmento de código:

```
int num , ct1= 0, ct2 = 0;
float alt, m1 = -1.0, m2 = -1.0;
scanf ("%d   %f ", &num,&alt);
while (num > 0)
{
  if (alt > m1)
     {
        m2 = m1;
        ct2 = ct1;
        m1 = alt;
        ct1 = 1;
     }
  else if (alt == m1)
           ct1 ++;
       else if (alt > m2)
                {
                   m2 = alt;
                   ct2 = 1;
                }
            else if (alt == m2)
                      ct2++;
   scanf ("%d  %f ",&num,&alt);
}
printf ( " %d   %d   %d   %d ",m1,ct1,m2,ct2);
```

Faça a simulação passo a passo deste segmento de código para os seguintes dados:

(10 1.75) (12 1.70) (22 1.70) (23 1.45) (32 1.75) (5 1.85) (4 1.70)
(16 1.75) (33 1.65) (36 1.75) (44 1.55) (38 1.85) (8 1.75) (0 1.90)

1- Funções

1.1-Desenvolva uma função para receber dois inteiros positivos a e b e verificar se b corresponde aos últimos dígitos de a. Por exemplo:

a	**b**	**Está Contido**
567890	890	Sim
1243	1243	Sim
2457	245	Não
4571	12457	Não

1.2- Dois números inteiros positivos com dois algarismos são pares combinados se forem pares e se os dígitos de um ocorrem na ordem inversa do outro. Por exemplo, os números 48 e 84 são pares combinados, enquanto os números 23 e 32 não são. Desenvolva uma função que recebe dois números inteiros positivos com dois dígitos e verifica se eles são pares combinados.

1.3- Desenvolva uma função que recebe as coordenadas de um ponto no plano cartesiano (X,Y). Retorne em que quadrante esse ponto se encontra. Se o ponto estiver no interior do quadrante retorne um número entre 1 a 4, se estiver sobre um dos eixos, retorne o valor -1, se estiver na origem retorne o valor zero.

1.4- Dizemos que um número inteiro n pode ser partido em dois números inteiros n1 e n2 se n pode ser escrito da forma n1n2, isto é, concatenar n1 com n2. Desenvolva uma função que recebe três inteiros positivos n, n1 e n2. Verifique se n pode ser partido em dois números inteiros n1 e n2. Vamos supor que esses números não contêm o dígito 0. Por exemplo: para n = 123456, n1 = 12 e n2 =3456, a resposta é sim, mas para n = 123456, n1 = 456 e n2 =123, a

2- Indução e Recursão

2.1- Prove que $2^n > 2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2^0$, para n > 1.

2.2- Prove que $1 + 2 + 2^2 + ... + 2^n = 2^{n+1} - 1$ para todo n≥ 0.

2.3- Prove que $1 + 3 + 6 + \ldots + \frac{n(n+1)}{2} = \frac{n(n+1)(n+2)}{6}$ para n ≥ 1.

2.4- Determine a relação de recorrência que expressa a seguinte sequencia:

2, 4, 8, 16, …

2.5- Determine a relação de recorrência que expressa a seguinte sequência:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

3- Recursão Linear

3.1- Dada a seguinte soma: 0 + 1 + 2 + 3 + 4 + ... + n

Desenvolva uma função recursiva e uma função iterativa para calcular os n primeiros termos.

3.2- Desenvolva uma função recursiva que recebe um número na numeração decimal e o converte para a numeração binária.

3.3- Desenvolva uma função recursiva que recebe como parâmetro um número real x e um número natural n, e calcula a potência de x elevado a n, com base na seguinte fórmula recorrente:

3.4- Desenvolva uma função recursiva que recebe um número inteiro positivo n e Calcula o valor da série Harmônica:

$$H_n = 1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + \ldots + 1/n$$

3.5- Faça a simulação da seguinte função recursiva. Suponha que n = 8

```
int func (int n )
{
     if ( n == 0 )  return 0;
     else  return n + func (n-1);
}
```

3.6- A função que descrevemos em seguida, calcula o MDC (m, n) pelo algoritmo de Euclides, para m e n estritamente positivos. Desenvolva uma função recursiva equivalente

```
int Euclides (int m, int n)
{
   int r;
   do
    {
      r = m % n;
      m = n;
      n = r;
    }
   while (r != 0);
   return m;
}
```

3.7- Desenvolva uma função recursiva que recebe um número inteiro positivo e retorna esse número na ordem inversa. Por exemplo, para n = 123, a sua função deve mostrar o número 321.

4- Vectores

4.1-Algoritmos Iterativos

4.1.1-Desenvolva uma função que recebe um vector com números reias e o número de elementos inseridos. Devolva o mesmo vector com os elementos na ordem inversa.

4.1.2-Desenvolva uma função que recebe um vector com números inteiros, o número de elementos inseridos, e dois números inteiro armazenados nas variáveis x e y. Inserir o elemento y após encontrar o primeiro elemento cujo conteúdo é igual a x.

4.1.3-Desenvolva uma função que recebe dois vectores com números inteiros e o número de registros inseridos em cada vector. Devolva num terceiro vector, a intercalação desses vectores. Por exemplo se A = { 1, 3, 5, 7 } e B = { 2, 4.1, 5, 6 } , a intercalação desses vectores é o vector C = { 1, 2, 3, 4, 5, 6, 7}

4.1.4-Desenvolva uma função que recebe um vector com números reais, o número de elementos inseridos, um número inteiro m > 0, e uma determinada posição k. Apagar, se for possível, incluíndo k, m posições consecutivas nesse vector. (utilize um ciclo apenas).

4.1.5- Um número é chamado de capicua se a sequência de algarismos do número lidos da esquerda para a direita for igual a sequência de algarismos lidos da direita para a esquerda. Por exemplo: os seguintes números são capicuas: 123454321, 54445, 789987, 121. Desenvolva uma função que recebe um vector com um número inteiro e o número de elementos inseridos nesse vector. Devemos salientar que cada elemento desse vector, contém um e apenas um algarismo. Verificar se esse número é do tipo capicua.

4.1.6-Desenvolva uma função que recebe um vector com números inteiros, o número de elementos inseridos e uma determinada chave. Eliminar se for possível, todas as ocorrências dessa chave.

4.1.7-Desenvolva uma função que recebe um vector com números inteiros entre 0 a 149, e o número de elementos inseridos. Devolva um vector, com os seguintes totais: no primeiro elemento, temos o total de ocorrências entre 0 a 4, no segundo elemento, o total de ocorrências entre 5 a 9, . . ., e no último elemento o total de ocorrências entre145 a 149.

4.1.8-Desenvolva uma função que recebe um vector com números inteiros possivelmente repetidos e o número de elementos inseridos. Retorne o número máximo de elementos consecutivos repetidos. Por exemplo, suponha que o vector possua os números {5,2,2,3,3,7,7,4,4,4,4,4,1,1}. A sua função deverá retornar o valor 5.

4.1.9-Desenvolva uma função que recebe dois vectores com números inteiros e o número de elementos inseridos em cada vector. Devolva num terceiro vector,

os elementos que não são comuns aos dois vectores. Para além disso, retorne o total de elementos inseridos nesse vector. Por exemplo, se o subprograma receber os conjuntos { 4, 3, 6, 1, 6 } e { 9, 0, 3, 1, 7, 6, 2 } deve devolver o conjunto { 4, 9, 0, 7, 2}.

4.1.10- Desenvolva uma função que recebe um vector com números inteiros e o número de elementos inseridos. Verifique se nesse vector os elementos de índice par contêm um conteúdo ímpar e vice-versa.

4.2- Algoritmos com Recursão Linear

4.2.1-Desenvolva uma função que entre outros argumentos, recebe um vector com números reias e o número de elementos inseridos. Retornar o número de elementos iguais a zero.

4.2.2- Desenvolva uma função que recebe entre outros argumentos um vector com números inteiros, o total de elementos inseridos, uma determinada chave e uma determinada posição no vector. Inserir essa chave nessa posição.

4.2.3-Desenvolva uma função que recebe entre outros argumentos dois vectores cujos elementos são 0 ou 1, e o número de elementos inseridos nos vectores. Vamos supor que os vectores possuem o mesmo número de elementos. Devolver num terceiro vector que contêm a soma binária desses números.

4.2.4-Desenvolva uma função que recebe entre outros argumentos um vector com números inteiros, o número de elementos inseridos e uma determinada chave. Verificar se essa chave encontra-se no vector.

4.2.5-Desenvolva uma função que recebe entre outros argumentos um vector com números reais e o número de elementos inseridos. Devolver o mesmo vector com os elementos na ordem inversa.

4.2.6-Desenvolva uma função que recebe entre outros argumentos dois vectores com números inteiros em ordem crescente e o número de elementos inseridos em cada vector. Intercalar esses elementos num terceiro vector de tal forma que não existem elementos repetidos.

4.2.7-Desenvolva uma função que recebe entre outros argumentos um vector com números inteiros e o número de elementos inseridos. Verifique se nesse vector os elementos de índice par contêm um conteúdo impár e vice-versa.

4.2.8-Um número é chamado de capicua se a sequência de algarismos do número lidos da esquerda para a direita for igual a sequência de algarismos lidos

da direita para a esquerda. Por exemplo: os seguintes números são capicuas: 123454321, 54445, 789987, 121. Desenvolva uma função que recebe um vector com um número inteiro e o número de elementos inseridos nesse vector. Devemos salientar que cada elemento desse vector, contém um e apenas um algarismo. Verificar se esse número é do tipo capicua.

4.2.9- Desenvolva uma função recursiva que recebe entre outros argumentos um vector com números inteiros, o número de elementos inseridos e uma determinada posição k. Separar esse vector em dois subvectores de tal forma que o elemento que está na posição k fique no primeiro vector.

4.2.10-Desenvolva uma função recursiva que recebe um vector com elementos do tipo real e número de elementos inseridos. Devolva o elemento máximo e o elemento mínimo.

4.3- Eliminação da Recursão

4.4- Cadeias de Caracteres da Recursão

4.4.1- Desenvolva uma função iterativa e uma função recursiva, para apagar de uma cadeia de caracteres passada como parâmetro, todas as letras minúsculas.

4.4.2- Desenvolva uma função iterativa e uma função recursiva, para substituir numa cadeia de caracteres passada como parâmetro, todas as letras do alfabeto pelo seu oposto, ou seja, a recebe z, b recebe x, etc.

4.4.3- Desenvolva uma função iterativa e uma função recursiva, para deslocar todos os caracteres de uma cadeia de caracteres para uma posição à direita e deslocar o último carácter dessa cadeia para a primeira posição do vector.

4.4.4- Desenvolva uma função iterativa e uma função recursiva, para concatenar os n primeiros elementos da cadeia de caracteres st1 na cadeia de caracteres st2.

4.4.5-Desenvolva uma função iterativa e uma função recursiva, para comparar os n primeiros dígitos das duas cadeias e retornar: devolver um número menor do que zero se a primeira cadeia é menor do que a segunda; 0 se as cadeias forem iguais; e um número positivo se a primeira cadeia é maior do que a segunda.

4.4.6-Desenvolva uma função iterativa, para verificar se uma cadeia está contida na outra. Essa função deve retornar 1 se a condição for verdadeira e 0 no caso contrário.

4.4.7-Desenvolva uma função iterativa e uma função recursiva, para eliminar todos os caracteres de branco de uma cadeia de caracteres e devolver essa cadeia.

4.4.8-Desenvolva uma função iterativa para remover todos os elementos repetidos de uma cadeia de caracteres e devolver essa cadeia.