---

1. **(a)** Qual é o resultado produzido por cada um dos seguintes programas?

```
program Teste1(output);
const C = 1.0;
var x: real;
begin x := C + 1;
      if x < 0 then
         x := 3
      else if x < 10 then
              x := x + 3
           else x := x - 3;
      writeln(x:4:1);
end.
```

```
program Teste2(output);
var n, m: integer;

procedure Muda(a: integer; var b: integer);
var c: integer;
begin c := a;
      a := b;
      b := c;
end;

begin n := 10; m:= 5;
      Muda(n, m);
      writeln(n, m);
end.
```

(b) Considerando `C` e `x` como no programa `Teste1`, diga, justificando, se a instrução a seguir é sintacticamente correcta.

```
C := 2*x*C + 1;
```

(c) Pretende-se armazenar a informação de 100 alunos utilizando a variável

```
var info: array [1 .. 100] of
            record Classificacao: (Mau, Medio, Bom, Excelente);
                   Presencas: integer;
            end;
```

Escreva uma instrução para atribuir a classificação de Bom ao aluno número 37.

(d) Compare as passagens de parâmetros por valor e por referência, explicando como funcionam e apresentando as suas diferenças.

(e) Exemplifique o método de ordenamento por selecção linear no vector `x = [30 5 3 17]` (`n = 4`).

---

2. Dados dois números naturais, a e b, a característica de $\log_b a$ é um valor inteiro. A característica de $\log_b a$ pode ser obtida dividindo a por b sucessivamente, até obter o número 0. Por exemplo, como $23 = 2^4 + 7$ a característica de $\log_2 23$ é 4 ($\log_2 23 = 4.523561956\ldots$).

Escreva um programa que leia a e b, com $a > b$, calcule a característica de $\log_b a$, utilizando apenas divisões e o processo descrito acima, e imprima o resultado.

3. Considere o seguinte tipo de dados

```
type Simbolo = ('A', 'C', 'G', 'T');
     Estrutura = array [1 .. 50] of Simbolo;
```

(a) Elabore um subprograma recursivo que verifique se uma dada variável do tipo `Estrutura` é simétrica relativamente ao elemento na posição central.

(b) Faça um subprograma que, dadas duas `Estrutura`s, `e1` e `e2`, construa uma nova `Estrutura`, `e3`, resultante do cruzamento de `e1` e `e2`. A nova variável é formada alternadamente por elementos de `e1` e de `e2`. Uma lista dada, de inteiros entre 1 e 50, indica os pontos em que os elementos passam a ser copiados a partir de outra `Estrutura`, sendo os primeiros de `e1`.

Por exemplo, para `e1 = AACGAATATT`, `e2 = ACCGAGTTTT` e as posições 2 e 5, deveria obter-se: `e3 = A|CCG|AATATT`.

(c) Escreva um programa que guarde duas sequências de 50 `A`s, `C`s, `G`s e `T`s dadas pelo utilizador em variáveis do tipo `Estrutura`. Deve igualmente ser armazenada uma sequência de posições das `Estrutura`s inserida pelo utilizador. A partir desta informação deve proceder-se ao cruzamento da primeira com a segunda `Estrutura`, se ambas forem simétricas, ou ao cruzamento da segunda com a primeira, caso contrário. A `Estrutura` resultante deve ser impressa no monitor.

Apresente apenas o código novo, substituindo as declarações dos subprogramas já desenvolvidos pelo respectivo cabeçalho e reticências.

1. **(a)** O programa `Teste1` produz: " 4.0".

O programa `Teste2` produz: "1010".

(b) Não é possível atribuir um valor a uma constante, portanto a instrução não é correcta.

(c) `Info[37].Classificacao := Bom`.

2.
```
program Ex2(input, output);
var a, b, copia, car: integer;
begin write('Insira a: ');
      readln(a);
      write('Insira b (<a): ');
      readln(b);
      copia := a;
      car := 0;
      while copia >= b do
      begin car := car + 1;
            copia := copia div b;
      end;
      writeln('caracteristica(', a, ',', b, ') = ', car);
end.
```

3. **(a)**
```
function Simetria(e: Estrutura; i, f: integer): boolean;
begin if i >= f then Simetria := true
      else if e[i] <> e[f] then Simetria := false
           else Simetria := Simetria(e, i+1, f-1);
end;
```

Esta versão pode ser melhorada através da utilização de recursão encapsulada, para evitar as cópias da `Estrutura` e em cada chamada recursiva da função.

(b)
```
procedure Cruzamento(e1, e2: Estrutura; bp: Lista; nbp: integer;
                     var e3: Estrutura);
var actuale, i, j: integer;
begin actuale := 1;
      i := 1;
      j := 1;
      while j <= nbp do
      begin while i < bp[j] do
            begin if actuale = 1 then e3[i] := e1[i]
                  else e3[i] := e2[i];
                  i := i + 1;
            end;
```

```
            j := j + 1;
            if actuale = 1 then actuale := 2
            else actuale := 1;
      end;
      while i <= 50 do
      begin if actuale = 1 then e3[i] := e1[i]
            else e3[i] := e2[i];
            i := i + 1;
      end;
end;
```

(c)

```
program Ex3(input, output);

type Simbolo = (A, C, G, T);
     Estrutura = array [1 .. 50] of Simbolo;
     Lista = array [1 .. 50] of integer;

var e1, e2, e3: Estrutura;
    bp: Lista;
    i, nbp: integer;

function Converte(l: char): Simbolo;
begin case l of
         'A': Converte := A;
         'C': Converte := C;
         'G': Converte := G;
         'T': Converte := T;
      end;
end;

function Inverte(l: Simbolo): char;
begin case l of
         A: Inverte := 'A';
         C: Inverte := 'C';
         G: Inverte := 'G';
         T: Inverte := 'T';
      end;
end;

procedure Ler(var e: Estrutura);
var i: integer;
    l: char;
begin write('Insira a estrutura: ');
      for i := 1 to 50 do
      begin read(l);
            e[i] := Converte(l);
```

```
      end;
      readln;
end;

function Simetria(e: Estrutura; i, f: integer): boolean;
...

procedure Cruzamento(e1, e2: Estrutura; bp: Lista; nbp: integer;
                     var e3: Estrutura);
...

procedure Escrever(e: Estrutura);
var i: integer;
begin write('Estrutura: ');
      for i := 1 to 50 do write(Inverte(e[i]));
      writeln;
end;

begin Ler(e1);
      Ler(e2);

      write('Insira o numero de breakpoints: ');
      readln(nbp);
      for i := 1 to nbp do read(bp[i]);

      if Simetria(e1, 1, 50) then Cruzamento(e1, e2, bp, nbp, e3)
      else Cruzamento(e2, e1, bp, nbp, e3);
      Escrever(e3);
end.
```