Fundação CECIERJ - Vice Presidência de Educação Superior a Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação I
AP2 1^o **semestre de 2016.**

---

1 Primeira questão (2 pontos)

Na AD2 foi apresentado um código recursivo para executar o casamento inexato de duas Strings. Porém, para mais de 11 caracteres o programa nunca chegava ao fim. Por que o uso de progração dinâmica resolve este problema?

Porque salva valores das chamadas recursivas já executadas numa tabela, evitando que uma mesma chamada recursiva seja executa mais de uma vez.

2 Segunda questão (2 pontos)

Desenhe uma interface para o lazarus que permita fornecer duas Strings, e exibir o número de movimentos necessários para casá-las e os movimentos executados. Forneça os nomes de todos os componentes utilizados.

Esta é a saída produzida pela função "reconstruct_path" no código abaixo.

"GAATTCAGTTA" → "GGATCGA" in 5 moves
M(G)S(A → G)M(A)D(T)M(T)M(C)D(A)M(G)D(T)D(T)M(A)

"thou−shalt−not" → "you−should−not" in 5 moves
D(t)S(h→y)M(o)M(u)M(-)M(s)M(h)I(o)S(a→u)M(l)S(t→d)M(-)M(n)M(o)M(t)

3 Terceira questão (4 pontos)

Considere o programa **strMatching** abaixo. Ele utiliza programação dinâmica e é exatamente igual ao que foi pedido na AD2. Pede-se:

1. Complete as três linhas omitidas após os sinais de atribuição, que atualiza o array auxiliar "opt" em (1).

2. Se o programa **strMatching** recebesse as Strings "cederj" e "cedrj", o que seria impresso?

   "cederj"− > "cedrj" in 1 moves
   M(c)M(e)M(d)D(e)M(r)M(j)

   Sugestão: considere os dois exemplos apresentados acima, onde "M" significa match, "D", delete, "I", insere, e "S", substitui.

3. O que a função "string_compare" deve retornar em (3)?

```
Program strMatching;

Type
  OperType = (UNDEFINED, MATCH, INSERT, DELETE);

cell = record
    /// custo desta celula
    cost: integer;
    /// celula pai
    parent: OperType;
end;

const
    MAXLEN = 50;

var str1, str2, str3, str4: String;
    dist: Integer;
    /// tabela de programacao dinamica
    m: Array [0..MAXLEN, 0..MAXLEN] of cell;

/// Retorna o custo de remover um caracter.
function indel (c: char): integer;
begin
    indel := 1;
end;

/// Retorna o custo de dois caracteres identicos,
/// como 0, e 1 caso contrario.
function char_match (c: char; d: char): Integer;
begin
    if ( c <> d ) then char_match := 1
    else char_match := 0;
end;

procedure row_init (i: integer);
begin
    m[0,i].cost := i;
    if ( i > 0 ) then m[0,i].parent := INSERT
    else m[0,i].parent := UNDEFINED;
end;

procedure column_init ( i: integer );
begin
    m[i,0].cost := i;
    if ( i > 0 ) then m[i,0].parent := DELETE
    else m[i,0].parent := UNDEFINED;
end;
```

```
procedure goal_cell (s: String; t: String; var i:integer; var j: integer
begin
    i := length(s);
    j := length(t);
end;

procedure insert_out (t: STring; j: integer );
begin
    write('I(', t[j], ')');
end;

procedure delete_out (s: String; i: integer );
begin
    write('D(', s[i], ')');
end;

procedure match_out (s: String; t: String; i: integer; j: integer );
begin
    if ( s[i] = t[j] ) then write('M(',s[i],')')
    else write('S(', s[i], '->', t[j], ')');
end;

function string_compare (s: String; t: String): integer;
var
    i, j: integer;
    k: OperType;
    opt: Array [OperType] of Integer;

begin
    for i := 0 to MAXLEN do
    begin
        row_init(i);
        column_init(i);
    end;
    for i := 1 to length(s) do
    begin
        for j := 1 to length(t) do
        begin
            opt[MATCH]  := m[i-1,j-1].cost + char_match(s[i],t[j]); (1)
            opt[INSERT] := m[i,j-1].cost   + indel(t[j]);
            opt[DELETE] := m[i-1,j].cost   + indel(s[i]);

            m[i,j].cost := opt[MATCH];
            m[i,j].parent := MATCH;
            for k := INSERT to DELETE do
```

```
                begin
                    if ( opt[k] < m[i,j].cost ) then
                    begin
                        m[i,j].cost := opt[k];
                        m[i,j].parent := k
                    end
                end
            end
        end;
        goal_cell(s,t,i,j);
        string_compare := m[i,j].cost      (3) <----------------------------
    end;
```

4 Quarta questão (2 pontos)

Escreva as três chamadas recursivas à função "reconstruct_path" que foram omitidas no código abaixo, para que ela imprima o caminho executado durante o processo de casamento.

```
procedure reconstruct_path (s: String; t: String; i: Integer; j: Integer );
begin
    if ( m[i,j].parent = UNDEFINED ) then exit;

    if ( m[i,j].parent = MATCH ) then
    begin
        reconstruct_path(s,t,i-1,j-1);  <----------------
        match_out(s,t,i,j);
        exit
    end;
    if ( m[i,j].parent = INSERT ) then
    begin
        reconstruct_path(s,t,i,j-1);      <----------------
        insert_out(t,j);
        exit
    end;
    if ( m[i,j].parent = DELETE ) then
    begin
        reconstruct_path(s,t,i-1,j);      <----------------
        delete_out(s,i);
        exit
    end
end;

procedure match_str (str1: String; str2: String);
begin
    dist := string_compare (str1, str2);
    writeln ( '"', str1, '" -> ', '"', str2, '" in ', dist, ' moves' );
    reconstruct_path(str1, str2, length(str1), length(str2));
    writeln;
    writeln;
end;

begin
    str1 := 'GAATTCAGTTA';
    str2 := 'GGATCGA';
    str3 := 'thou-shalt-not';
    str4 := 'you-should-not';
    match_str (str1, str2);
    match_str (str3, str4);
    if ParamCount > 1 then begin
            writeln ( ParamStr(0), ' ', ParamStr(1), ' ', ParamStr(2));
            match_str (ParamStr(1), ParamStr(2));
    end
end.
```