**Avaliação – Analista programador**

Nome completo: Ricardo Takemura

Disponibilidade para início: Imediato

**Teste de conhecimentos técnicos**

As questões 1, 2 e 3 são obrigatórias. Escolha 1 ou mais questões entre as questões 4, 5 e 6.

1. Utilizando PORTUGUÊS ESTRUTURADO, resolva o seguinte problema, otimizando ao máximo a solução:  
   Receba n valores numéricos e exiba o maior deles.

|  |  |
| --- | --- |
| Exemplo de entrada | Exemplo de saída |
| n = 5 {4, 6, 5, 20, 7} | 20 |

função maior\_valor(valores: Lista<Inteiro>): Inteiro

var maximo: Inteiro;

inicio

maximo = Inteiro.minimo;

para i = 1 até tamanho(valores) faça

se maximo < valores[i] então maximo = valores[i];

retorna maximo;

fim

1. Ainda usando PORTUGUÊS ESTRUTURADO, altere o algoritmo do problema anterior, para além de devolver o maior valor, devolver o **segundo** maior valor.

|  |  |
| --- | --- |
| Exemplo de entrada | Exemplo de saída |
| n = 5 {4, 20, 6, 5, 7} | Primeiro: 20 Segundo: 7 |

função maior\_valores(valores: Lista<Inteiro>): Lista<Inteiro>

var maximo: Lista<Inteiro>;

inicio

máximo = {Inteiro.minimo, Inteiro.minimo};

para i = 1 até tamanho(valores) faça

se maximo[1] < valores[i] então

inicio

maximo[2] = maximo[1];

maximo[1] = valores[i];

fim;

retorna maximo;

fim

1. Troque o valor entre duas variáveis numéricas sem utilizar uma variável auxiliar, usando apenas operações matemáticas básicas.

|  |  |
| --- | --- |
| Exemplo de entrada | Exemplo de saída |
| var a = 17; var b = 13; | a = 13 b = 17 |

procedimento troca(var a: Inteiro, var b: Inteiro)

inicio

a = a + b;

b = a - b;

a = a - b;

fim

1. Suponha que você precisa fazer um programa que devolva o valor do troco de uma compra.   
   Suponha agora que você tenha um conjunto de moedas M = { M1, M2, ..., Mn | M1 > M2 > Mn } e você possui uma quantidade infinita dessas moedas. Dado um valor V, descreva um algoritmo que devolva a quantidade mínima de moedas a serem utilizadas para formar esse valor. Observe que você não precisa descrever a quantidade de cada tipo de moeda, apenas a quantidade mínima que forma o troco.

|  |  |
| --- | --- |
| Exemplo de entrada | Exemplo de saída |
| M = {25, 15, 5} V = 30 | 2 moedas |

função quantidade\_moedas(var valor: Inteiro, var moedas: Lista<Inteiro>, var contador: Inteiro)

var troco: Inteiro;

var i, j: Inteiro;

var contador, min\_contador: Inteiro;

início

min\_contador = Inteiro.maximo;

para i = 1 até tamanho(moedas)

inicio

contador = 0;

troco = 0;

j = i;

faça

se valor >= moedas[j] então

início

troco = valor - moedas[j];

contador = contador + 1;

fim;

senão j = j + 1;

até que troco = 0 ou j > tamanho(moedas);

se contador > 0 e contador < min\_contador então min\_contador = contador;

fim;

retorna min\_contador;

fim

1. Existe uma maneira de saber quão próximo um array está de estar completamente ordenado usando a técnica de contar as inversões em um array.   
   Dois elementos array[i] e array[j] formam uma inversão se array[i] > array[j] e i < j, onde i e j são índices do array.   
   Faça um algoritmo que conte a quantidade de inversões contidas em um array.

|  |  |
| --- | --- |
| Exemplo de entrada | Exemplo de saída |
| M = {2, 4, 1, 3, 5} | 3 (2, 1), (4, 1), (4, 3) formam inversões (seu algoritmo não precisa listá-los) |

função quantidade\_inversoes(var lista: Lista<Inteiro>): Inteiro

var contador: Inteiro;

início

contador = 0;

para i = 1 até tamanho(lista) faça

para j = i + 1 até tamanho(lista) faça

se lista[i] > lista[j] então contador = contador + 1;

retorna contador;

fim

1. Suponha que você precise fazer um programa de reserva de salas.  
   Dado um conjunto de reservas com horário de início e término M = {(ini1, ter1), (ini2, ter2), ... } ordenados pela hora de início, faça um algoritmo que calcule a quantidade mínima de salas para atender todas as reservas.

|  |  |
| --- | --- |
| Exemplo de entrada | Exemplo de saída |
| M = {(8:00, 12:00), (10:00, 11:00), (13:00, 20:00), (13:00, 15:00), (16:00, 18:00)} | 2 |

estrutura Periodo

var horario\_inicial;

var horario\_final;

fim;

função quantidade\_minima\_salas(var periodos: Lista<Periodo>)

var salas: Matriz<Periodo>;

var i, sala, horario: Inteiro;

var encaixou: Boleano;

início

para i = 1 até tamanho(periodo) faça

início

encaixou = falso;

se vazio(salas) faça

início

salas[1][1] = periodo[i];

encaixou = verdadeiro;

fim;

senão

início

sala = 1;

faça

horario = 1;

faça

se salas[sala][horario].horario\_final <= periodo[i].horario\_inicial então

início

encaixou = verdadeiro;

salas[sala][horario + 1] = periodo[i];

fim;

horario = horario + 1;

até que encaixou = verdadeiro ou horario > tamanho(salas[sala]);

sala = sala + 1;

até que encaixou = verdadeiro ou sala > tamanho(salas);

salas[sala][horario] = periodo[i];

encaixou = verdadeiro;

fim;

fim;

retorna tamanho(salas);

fim