E-fólio Global | Instruções para a realização do E-fólio

UNIDADE CURRICULAR: LINGUAGENS DE PROGRAMAÇÃO

CÓDIGO: 21077

DOCENTE: RICARDO BAPTISTA

A preencher pelo estudante

NOME: Francisco José Pinto de Amaral

N.º DE ESTUDANTE: 1802876

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 07/06/2022

TRABALHO / RESOLUÇÃO:

[GRUPO I]

1. a)

```
(* Apresentação das listas não vazias e tamanhos diferentes*)

let lista1 = [2;1;4;6;3];;
let lista2 = [1;1;4;5;6;8;2];;

(*Comparação de valores da lista*)

let compara l1 l2 =
   if l1 = l2 then 0
   else if l1 > l2 then 1
   else if l1 < l2 then -1
   else []
;;

(*Apresentação da nova lista com o retorno na comparação de
valores *)

let getListaCompara list1 list2 =
   List.compara list1 list2;;

getListaCompara

Resultado=[1;0;0;1;-1;([]);([])]
```

1.b)

```
 /*Para o caso das listas estarem vazias*/compara([],[],[],[],[]).

/*Comparação de valores das listas*/
compara([X1|Y1],[X2|Y2],[X3|Y3],[X4|Y4],[X5|Y5]):-
    X1 == X2,
    X3 == X4,
    X5 is 0,
    compara(Y1, Y2, Y3, Y4, Y5).

compara([X1|Y1],[X2|Y2],[X3|Y3],[X4|Y4],[X5|Y5]):-
    X1 > X3,
    X2 > X4,
    X5 is 1,
    compara(Y1, Y2, Y3, Y4, Y5).

compara([X1|Y1],[X2|Y2],[X3|Y3],[X4|Y4],[X5|Y5]):-
    X1 < X2,
    X3 < X4,
    X5 is -1,
    compara(Y1, Y2, Y3, Y4, Y5).

/*Simulação de duas listas, comparação e apresentação de
resultados*/
teste :-
    P = [2,1,4,6,3],
    Q = [1,1,4,5],
    compara(P, Q, N),
    write("Apresentação da nova lista:"), nl,
    write(N).

teste.
```

1. c)

```
public class Teste_1 {

    public static ArrayList<Integer> compara(ArrayList<Integer>
lista_1, ArrayList<Integer> lista_2){
        ArrayList<Integer> lista_r = new ArrayList<>();
        if (lista_1.size()==lista_2.size()) {
            for (int i=0; i<lista_1.size();i++){
                if (lista_1.get(i) > lista_2.get(i))
                    lista_r.add(1);
                else
                    if (lista_1.get(i) == lista_2.get(i))
                        lista_r.add(0);
                    else
                        lista_r.add(-1);
            }
        }
        return lista_r;
    }

    public static void main(String[] args) {
        ArrayList<Integer> l1 = new ArrayList<>();
        ArrayList<Integer> l2 = new ArrayList<>();
        ArrayList<Integer> res;
        // primeira lista para teste
        l1.add(3);
        l1.add(40);
        l1.add(50);
        // segunda lista para teste:
        l2.add(30);
        l2.add(40);
```

```
        l2.add(5);

        // testa funcao compara (listas com mesmo tamanho):
        res=compara(l1, l2);
        System.out.println("Resultado Primeira comparacao\n(listas c/
mesmo numero de eementos):");
        if (res.size()>0)
            for (int i=0; i<res.size();i++)
                System.out.println(res.get(i));
        else
            System.out.println("Lista vazia\n");

        // acrescenta um elemento a uma das listas
        l2.add(7);

        // testa funcao compara (listas com tamanhos diferentes):
        res=compara(l1, l2);
        System.out.println("\nResultado Segunda comparacao\n(listas
c/ diferenteo numero de eementos):");
        if (res.size()>0)
            for (int i=0; i<res.size();i++)
                System.out.println(res.get(i));
        else
            System.out.println("Lista vazia\n");
    }
}
```

[GRUPO II]

1.

(*Criação da árvore*)

```
type 'a b_tree =
    | Empty
    | Node of 'a * 'a b_tree * 'a b_tree
    ;;
```

(*Definição da árvore desordenada*)

```
let tree =
  Node(1,
  Node(5, Node(10, Empty, Empty), Node(15, Empty, Empty)),
  Node(7, Node(2, Empty, Empty), Node(6, Empty, Empty))
  ) ;;
```

(*Organização e Apresentação dos dados*)

```
let rec order t =
  match t with
  | Empty -> 0
    | Node(l,v,r)->
            print_string (string_of_int v);print_string "->";
            (order l);
            (order r)
;;

let()= order tree;;
```

2.

```
/*Fatos - Base de conhecimento*/
ProdutosAlimentares(leite,lacticinios,24,0.6,0.8).
ProdutosAlimentares(twix,doces,6,4.6,9.4).
ProdutosAlimentares(amendoas,doces,600,0.4,0.8).
ProdutosAlimentares(chocolatePreto,doces,34,1.2,1.7).
ProdutosAlimentares(cerveja,bebidas,80,1.1,1.3).
ProdutosAlimentares(vinho,bebidas,49,9.6,18.1).
ProdutosAlimentares(queijo,lacticinios,12,5.8,7.4).
ProdutosAlimentares(agua,bebidas,130,0.3,0.6).

/*Calculo da margem de lucro*/
Margem_Lucro_Categorias:-
    findall(X,ProdutosAlimentares(_,_,_,X,_),LCompra),
    findall(Y,ProdutosAlimentares(_,_,_,_,Y),LVenda),
    M > 0.1
    -> M is (1 - (LCompra / LVenda)), write(M);
        True.

/*predicado para apresentação da lista ordenada da margem de
forma crecente*/
sort_list:-
    findall(Z,ProdutosAlimentares(_,Z,_,_,_),LCat),
    sort(M, Lsort),
    write(Lcat,Lsort).
```

3. a) e b)

```
public class Pessoa {

    private String nome;

    private String dataNasc;

    private String contacto;

    private String email;


  // Constructor

    public Pessoa(String nome, String dataNasc, String contacto,
String email){

        this.nome = nome;

        this.dataNasc=dataNasc;

        this.contacto=contacto;

        this.email=email;

    }

  // Metodos Setters e Getters

    public String getNome() {

        return nome;

    }

    public void setNome(String nome) {

        this.nome = nome;
```

```
}
public String getDataNasc() {
    return dataNasc;
}
public void setDataNasc(String dataNasc) {
    this.dataNasc = dataNasc;
}
public String getContacto() {
    return contacto;
}
public void setContacto(String contacto) {
    this.contacto = contacto;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}


// Metodos para imprimir todos os dados de uma pessoa:
```

```
    public void imprimeDadosPessoaLin(){

        System.out.format("%20s                %10s                %12s
%25s",nome,dataNasc,contacto,email);

    }

}


import java.util.List;

public class Agenda {

    private Pessoa pessoa;

    private boolean status;

    private String dataInclusao;


    private List<Agenda> listaPessoas;


  //Constructor:

    public Agenda(){

        listaPessoas = null;

        status = false;

        dataInclusao = "";

    }
```

```
// Metodos Setters e Getters

  public Pessoa getPessoa() {

      return pessoa;

  }

  public void setPessoa(Pessoa pessoa) {

      this.pessoa = pessoa;

  }

  public boolean getStatus() {

      return status;

  }

  public void setStatus(boolean status) {

      this.status = status;

  }

  public String getDataInclusao() {

      return dataInclusao;

  }

  public void setDataInclusao(String dataInclusao) {

      this.dataInclusao = dataInclusao;

  }


// Metodo para inserir Pessoa na Agenda
```

```
    public void inserirPessoa (String nome, String dataN, String
contacto, String email, String dataIn, boolean status) {

        Agenda novaAg = new Agenda ();

        Pessoa novaPes = new Pessoa(nome, dataN, contacto,email);

        novaAg.pessoa=novaPes;

        novaAg.status=status;

        novaAg.dataInclusao=dataIn;

        listaPessoas.add(novaAg);

    }


  // Metodo para listar Pessoa da Agenda:

    public void listarPessoas(){

        List<Agenda> aux = listaPessoas;



        for(int i=0; i<listaPessoas.size(); i++){

           if (listaPessoas.get(i).status)

                listaPessoas.get(i).pessoa.imprimeDadosPessoaLin();

        }

    }

}
```