

**UNIDADE CURRICULAR:** LINGUAGENS DE PROGRAMAÇÃO

**CÓDIGO:** 21077

**DOCENTE:** RICARDO BAPTISTA

**A preencher pelo estudante**

**NOME:** Hernani Filipe Resendes Coelho

**N.º DE ESTUDANTE:** 1800045

**CURSO:** Engenharia Informática

**DATA DE ENTREGA:** 07/06/2022

**TRABALHO / RESOLUÇÃO:**

1a)

let *lista1* **=** **[**5;2;-4;5**]**;;

let *lista2* **=** **[**5;4;6;-8;1**]**;;

let rec *funcao* *lista1* *lista2* **=** (

  match (lista1, lista2) with

**|[]**,\_ -> **[]**

**|**\_**,[]** **->** **[]**

**|**(*h1*::*t1*)**,**(*h2*::*t2*) **->** if (h1 = h2) then 0::funcao t1 t2 else if (h1>h2) then 1::funcao t1 t2 else -1::funcao t1 t2

);;

let *l1* **=** List**.**length lista1;;

let *l2* **=** List**.**length lista2;;

if(l1 = l2) then funcao lista1 lista2 else [];;

1b)

%base conhecimento

funcao([], [], []).

%Regra

funcao([H1|T1], [H2|T2], [HR|TR]) :-

    length([H1|T1], L1), length([H2|T2], L2), L1=L2, true,

    ((H1 = H2), HR is 0, funcao(T1, T2, TR);

    (H1 > H2), HR is 1, funcao(T1, T2, TR);

    (H1<H2), HR is -1, funcao(T1, T2, TR)).

%query

funcao([1,2,-4,5], [5,4,6,-8], X).

1c)

import java.util.\*;

public class Main {

    public static void main(String[] args) {

      List<Integer> lista1 = Arrays.asList(1,6,3,4,5,1);

      List<Integer> lista2 = Arrays.asList(5,4,3,5,7);

      List<Integer> lista3 = new ArrayList<Integer>();

    if(lista1.size() == lista2.size()){

        for(int i=0; i < lista1.size(); i++){

            if(lista1.get(i) == lista2.get(i))

                lista3.add(0);

            else if(lista1.get(i) > lista2.get(i))

                lista3.add(1);

            else

                lista3.add(-1);

        }

        for(int i=0; i < lista3.size(); i++){

            System.out.print(" " + lista3.get(i));

          }

    }

    else System.out.print("lista vazia []");

    }

}

Grupo II

1)

type *'a* **binary\_tree** **=**

**|** **Empty**

**|** **Node** of *'a* **\*** *'a* **binary\_tree** **\*** *'a* **binary\_tree**;;

let *example\_tree* **=**

**Node** (1, **Node** (2, **Node** (4, **Empty**, **Empty**), **Node** (5, **Empty**, **Empty**)), **Node** (3, **Empty**, **Node** (6, **Node** (7, **Empty**, **Empty**), **Empty**)));;

//Por falta de tempo faltou definir a funcao responsável por ordenar os valores das folhas de forma descendente

2)

%Base conhecimento

produtosAlimentares(maca, fruta, 10, 1, 2).

produtosAlimentares(leite, laticinios, 3, 3, 4).

produtosAlimentares(pao, padaria, 100, 0.1, 0.2).

produtosAlimentares(pera, fruta, 100, 1.5, 2.5).

%Regra

margem\_Lucro\_Categorias(MargemMinima, L) :-

    findall(Categoria, produtosAlimentares(Nome,Categoria,Stock,PrecoCompra,PrecoVenda), L),

    Sort(Lista, Lista1).

%Não tive tempo de implementar como queria o predicado solicitado, pelo que este não está funcional

3)

import java.util.\*;

import java.util.Date;

public class Main {

    //classe pessoa

    public static class Pessoa {

        private String nome;

        private Date dataNascimento;

        private String contacto;

        private String email;

        //Gets

        public String getNome() {

            return nome;

        }

        public Date getDataNascimento() {

            return dataNascimento;

        }

        public String getContacto() {

            return contacto;

        }

        public String getEmail() {

            return email;

        }

        //Sets

        public void setNome(String nome) {

            this.nome = nome;

        }

        public void setDataNascimento(Date data) {

            this.dataNascimento = data;

        }

        public void setContacto(String contacto) {

            this.contacto = contacto;

        }

        public void setEmail(String email) {

            this.email = email;

        }

        //Construtor de pessoa

        public Pessoa(String nome, Date dataNascimento, String contacto, String email){

            this.nome = nome;

            this.dataNascimento = dataNascimento;

            this.contacto = contacto;

            this.email = email;

        }

        //Metodo imprimir pessoa

        public void ImprimirPessoa(List<Pessoa> pessoas){

               for (Pessoa pessoa : pessoas) {

                System.out.print(pessoa.getNome() + " " + pessoa.getDataNascimento() + " "

                                 + pessoa.getContacto() + " " + pessoa.getEmail());

            }

        }

    }

    //classe agenda

    public static class Agenda {

        private Pessoa pessoa;

        private boolean status;

        private Date inclusao;

        private List<Agenda> lAgenda;

        //Gets

        public Pessoa getPessoa() {

            return pessoa;

        }

        public boolean getStatus() {

            return status;

        }

        public Date getInclusao() {

            return inclusao;

        }

        //Sets

        public void setPessoa(Pessoa pessoa) {

            this.pessoa = pessoa;

        }

        public void setStatus(boolean status) {

            this.status = status;

        }

        public void setInclusao(Date inclusao) {

             this.inclusao = inclusao;

        }

        //Construtor por default

        public Agenda(){}

        //Metodo AdicionarPessoa

        public void adicionarPessoa(){

            setPessoa(this.pessoa);

            setStatus(status);

            setInclusao(inclusao);

            lAgenda.add(this);

        }

        //Metodo imprimir agenda

        public void imprimirPessoaAgenda(List<Pessoa> pessoas, List<Agenda> agenda){

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

                Pessoa pessoa = pessoas.get(i);

                for (int j = 0; j < agenda.size(); j++){

                    if(agenda.get(j).pessoa == pessoa && agenda.get(j).status == true)

                    System.out.println(pessoa.getNome() + " " + pessoa.getDataNascimento() + " "

                                        + pessoa.getContacto() + " " + pessoa.getEmail());

                    }

            }

        }

    }

}