```
##############################          Grupo 1 OCAML
############################################################

(* Definição do tipo lista *)
(* type 'a list = [] | :: of 'a * 'a list *)

(* Criterios
    1 - Indicação das listas e seus dados (ainda que usando dados
parciais do enunciado)
    2 - Função Soma sob paradigma funcional
    3 - Aplicação de função soma nos elementos de duas ou mais listas
    4 - Função ou impressão do resultado (ou cuidado com a apresentação
dos resultados)
    5 - Demonstração dos resultado (usando os dados do enunciado ainda
que parciais)
*)

(* Declaracao das  listas de inteiros dos empregados *)
let a = [8;7;6]
let b = [8;7;8]
let d = [8;8;8]
let c = [8;8;0]
let e = [8;8;8]

(* Definiçao da funçao soma *)
let soma a1 a2 =
    a1 + a2 ;;

(* Aplicação da função às listas *)
let report a b c d e =
    let ab = List.map2 soma a b in
    let dc = List.map2 soma d c in
    let abdc = List.map2 soma ab dc in

    print_newline();
    print_string " O resultado total das horas";
    print_newline();
    List.map2 soma abdc e;;

report a b c d e ;


##############################          Grupo 1 PROLOG
############################################################


/*
Criterios
    1 - IndicaÃ§Ã£o das listas e seus dados (ainda que usando dados
parciais do enunciado)
    2 - Predicado(s) com loÌgica da soma (sob paradigma de
programacÌ§aÌƒo loÌgica e com paragem (cut))
    3 - Predicado ou impressÃ£o do resultado (ou cuidado com a
apresentaÃ§Ã£o dos resultados)
*/

/* Horas por dia nas cinco listas e retorna na Lret */

/* LÃ³gica de soma dos elementos da lista */
somaHoras([], [], [], [], [], Lret ) .
```

```
somaHoras([X1|T1], [X2|T2], [X3|T3], [X4|T4], [X5|T5], [XT|TT] ) :-
    XT is X1 + X2 + X3 + X4 + X5,
    somaHoras(T1, T2, T3, T4, T5, TT).


/* LÃ³gica para a apresentaÃ§Ã£o dos resultados*/

/* Pode ser completado ... */
run :-
    write('Inicia analise'),
    nl,
    La = [8,7,6],
    Lb = [8,7,8],
    Ld = [8,8,8],
    Lc = [8,8,0],
    Le = [8,8,8],
    somaHoras(La, Lb, Lc, Ld, Le, Lret),
    write("resultado final"),
    nl,
    write(Lret).

run.
```

```
##############################          Grupo 1 JAVA
############################################################


import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;


/*
Critérios:
    1 - Indicação dos campos/propriedades
    2 - Inicialização dos campos
    3 - Método de cálculo das horas
    4 - Método ou impressão do resultado (demonstração de resultados)
 */


public class grupoI1questao1c {
    // Campos da classe: 5 listas, uma por empregado

    ArrayList<Integer> A ;
    ArrayList<Integer> B ;
    ArrayList<Integer> C ;
    ArrayList<Integer> D ;
    ArrayList<Integer> E ;

    //Contrutores
    public grupoI1questao1c(){
        A= B = C = D = E = new ArrayList<>();
    }
```

```
    public grupoI1questao1c(ArrayList<Integer> iA, ArrayList<Integer> iB,
ArrayList<Integer> iC, ArrayList<Integer> iD, ArrayList<Integer> iE ){
        A = iA;
        B = iB;
        C = iC;
        D = iD;
        E = iE;
    }

    // Método auxiliar para cálculo das horas
    public ArrayList<Integer> getHorasPorDia(){
        ArrayList<Integer> result_HorasPorDia = new ArrayList<>();
        int sumAux=0;

        if (A.size() == B.size() && A.size() == C.size() && A.size() ==
D.size() &&
                A.size() == E.size()){
            for (int i=0; i< A.size(); i++){
                sumAux = A.get(i) + B.get(i) + C.get(i) + D.get(i) +
E.get(i);
                result_HorasPorDia.add(sumAux);
            }
        }else{
            result_HorasPorDia = null;
        }

        return(result_HorasPorDia);
    }

    // Método auxiliar de impressão
    public void imprime(ArrayList<Integer> iL){

        for (int i=0;i<iL.size();i++){
            System.out.print("\t" +i);
        }
        System.out.println("");
        for (int i=0;i<iL.size();i++){
            System.out.print("\t" + iL.get(i));
        }
        System.out.println("\n");
    }


    // Método principal
    public static void main(String []args){
        // Inicializa os dados com exemplos do enunciado até ao dia 3.

        // Não era preciso exemplificar  todos, bastava 1...
        ArrayList<Integer> A = new
ArrayList<Integer>(Arrays.asList(8,7,6));
        ArrayList<Integer> B = new
ArrayList<Integer>(Arrays.asList(8,7,8));
        ArrayList<Integer> C = new
ArrayList<Integer>(Arrays.asList(8,8,0));
        ArrayList<Integer> D = new
ArrayList<Integer>(Arrays.asList(8,8,8));
        ArrayList<Integer> E = new
ArrayList<Integer>(Arrays.asList(8,8,8));

        grupoI1questao1c exer1c = new grupoI1questao1c(A, B, C, D, E);
```

```
        exer1c.imprime(exer1c.getHorasPorDia() );
    }
}
```

```
##############################           Grupo 2 OCAML
#############################################################


(*
Criterios
1 - definião de árvore binária
2 - Inicialização de uma árvore com dados
3 - Cálculo do número primo
4 - Apresentação dos resultados (só numeros primos)
*)

(* Definição da árvore binária *)
type 'a tree =
  Nil
  | Node of 'a * 'a tree * 'a tree ;;

let arvore = Node (1, Node(2, Node(4, Nil, Nil), Nil),
                      Node(3, Node(7, Nil, Nil), Nil)) ;;

(* Função para verificar se é primo *)
let is_prime n =
    let n = abs n in
    let rec is_not_divisor d =
      d * d > n || (n mod d <> 0 && is_not_divisor (d+1)) in
    n <> 1 && is_not_divisor 2;;

(* Aplica função à árvore *)
let rec check_tree tr =
  match tr with
    | Nil -> []
    | Node(x, l, r) -> if is_prime x then x::check_tree l@ check_tree r
                       else check_tree l@ check_tree r;;

check_tree arvore;;
```

```
##############################           Grupo 2 PROLOG
#############################################################


/*
Grupo II Exercicio 2

CritÃ©rios
1 - definiÃ§Ã£o e apresentaÃ§Ã£o de predicados desempenho (pelo menos 2
projetos)
2 - IndicaÃ§Ã£o de lÃ³gica para cÃ¡lculo de projeto com mais horas de
trabalho
```

```
3 - Predicado projectos de mais horas de trabalho (em Lista ordenada)
4 - ApresentaÃ§Ã£o dos resultados
*/


desempenho(funcA, proja, 40, 0).
desempenho(funcB, projb, 40, 0).
desempenho(funcC, proja, 20, 0).
desempenho(funcD, projb, 10, 0).

/* Os Dias de Folga podem ser ignorados */
sum_list([], 0).
sum_list([H|T], Sum) :-
   sum_list(T, Rest),
   Sum is H + Rest.

find_hours_proj([]).

find_hours_proj([[]|[]]).

find_hours_proj([X|[]]):-
    findall(H1, desempenho(_,X,H1,_), Lhor0),
    sum_list(Lhor0, Sum0),
    write([X,Sum0]).


find_hours_proj([X|List]):-
  findall(H, desempenho(_,X,H,_), Lhor),
  sum_list(Lhor, Sum),
  write([X,Sum]),
  find_hours_proj(List).

/* create association list */
sort_list :-
  findall(P, desempenho(_,P,_,_), L),
  sort(L, Lsort),
  write(Lsort),
  find_hours_proj(Lsort).


  ##############################          Grupo 2 JAVA
############################################################


  CLASS CONCORRENTE



/*
Critérios:
    1 - Inclui todos as propriedades (nome, email, telefone, face id )
    2 - Encapsula ou indica métodos de encapsulamento para propriedades
(get e sets)

 */

/*
Classe para representar o Concorrente
 */
public class Concorrente implements Comparable<Concorrente>{
```

```
    private String nome;
    private String email;
    private int telefone;
    private int face_id;

    /* Estes campos podiam ficar numa classe distinta (mas simplifica-se
e
    assume-se que cara concorrente só pode ter uma mensagem a concurso */
    private String post;
    private int nlikes;

    /* Construtor (não era necessário) */
    public Concorrente(String nome, String email, int telefone, int
face_id, String post, int nlikes) {
        this.nome = nome;
        this.email = email;
        this.telefone = telefone;
        this.face_id = face_id;
        this.post = post;
        this.nlikes = nlikes;
    }

    /* Método para nos permitir comparar das mensagens de dois
concorrents
    *
    * Retorna:
    *  -> 1 caso o Nlikes seja maior
    *  -> 0 em que é igual
    *  -> -1 em é menor.
    *  *
    *  (segue a lógica do Java implementada noutras classes tipo Integer)
    * */
    @Override
    public int compareTo(Concorrente o) {
        if (this.getNlikes() > o.getNlikes()){
            return 1;
        }else {
            if (this.getNlikes() == o.getNlikes())
                return 0;
            else
                return -1;
        }
    }

    /* Para facilitar a impressão  */
    @Override
    public String toString(){
        String aux = "";
        aux += "Nome: " + this.getNome();
        aux += "  | Mensagem: " + this.getPost();
        aux += "  | nLikes: " + this.getNlikes();
        return aux;
    }


    /* Gets e Sets */
    public String getNome() {
        return nome;
    }
```

```
    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public int getTelefone() {
        return telefone;
    }

    public void setTelefone(int telefone) {
        this.telefone = telefone;
    }

    public int getFace_id() {
        return face_id;
    }

    public void setFace_id(int face_id) {
        this.face_id = face_id;
    }

    public String getPost() {
        return post;
    }

    public void setPost(String post) {
        this.post = post;
    }

    public int getNlikes() {
        return nlikes;
    }

    public void setNlikes(int nlikes) {
        this.nlikes = nlikes;
    }



}
```

CLASS CONCURSO

```
/*
Critérios 3a:
    3 - Define métodos: seleçãoVencedores, inscriçãoConcorrentes,
analisaGostoMsgs (e outros auxiliares)
```

```
Critérios 3b:
    1- Selecção correta dos vencedores
    2- apresenta os três premiados
    3- Ordena vencedores
 */

import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;

/* Classe que representa um concurso */

public class Concurso {


    ArrayList<Concorrente> concorrentes ;


    public Concurso() {
        this.concorrentes = new ArrayList<Concorrente>();
    }

    /* Adicionar os concurrentes a um concurso
    * Adicionar à lista um concorrente.
    *
    * Return false em caso de erro de inserção
    *
    * alinea 3a
    * */
    public boolean inscricaoConcorrente(Concorrente aux){
        if (concorrentes.add(aux))
            return true;
        else
            return false;
    }

    /*
    * Metodo para analisar as mensagens e o numero de likes.
    *
    * Retorna a lista ordenada por numero de likes (ordem decrescente)
    *
    * alinea 3a e 3b
    * */
    public ArrayList<Concorrente> analisaGostoMsgs(){
        ArrayList<Concorrente> aux = concorrentes;

        Collections.sort(aux, Collections.reverseOrder());

        return aux;
    };

    /*
     * Método para apresentar os 3 primeiros
     *
     * alinea 3a e 3b
     * */
    public void selecaoVencedores(){
        int posicoes = 3;
        int conta = 1;
```

```
        ArrayList<Concorrente> concorrentes_ord =
this.analisaGostoMsgs();

        for (Concorrente ic : concorrentes_ord){
            if (conta <= posicoes){
                System.out.println("" + conta + " lugar --> " + ic);
            }else { break;}
            conta +=1;
        }
    }

    // Exemplo não era necessário no p-folio
    public static void main (String [] args){

        Concorrente b1 = new Concorrente("pat 1", "par1@gmail.com",
111222333, 1, "O meu post vencedor", 10);
        Concorrente b2 = new Concorrente("pat 2", "par2@gmail.com",
111222333, 2, "O meu post 2 ", 2);
        Concorrente b3 = new Concorrente("pat 3", "par3@gmail.com",
111222333, 3, "O meu post 3", 1);
        Concorrente b4 = new Concorrente("pat 4", "par4@gmail.com",
111222333, 4, "O meu post 4", 5);

        Concurso concurso = new Concurso();

        /* Inscrição */
        concurso.inscricaoConcorrente(b1);
        concurso.inscricaoConcorrente(b2);
        concurso.inscricaoConcorrente(b3);
        concurso.inscricaoConcorrente(b4);

        /* apresenta Selecção*/
        concurso.selecaoVencedores();


        /*
        Deve imprimir algo do género:

        1 lugar --> Nome: pat 1  | Mensagem: O meu post vencedor  |
nLikes: 10
        2 lugar --> Nome: pat 4  | Mensagem: O meu post 4  | nLikes: 5
        3 lugar --> Nome: pat 2  | Mensagem: O meu post 2   | nLikes: 2

         */
    }

}
```