TRABALHO / RESOLUÇÃO:

Grupo I

1a) O'Caml;

```
let rec media a b =
  match (a, b) with
  | [], _ -> b
  | _, [] -> a
  | (hd :: tl), (hd2 :: tl2) -> if (hd > 0 && hd2 > 0)
      then (hd+hd2)/2 :: (media tl tl2)
      else hd + hd2 :: (media tl tl2);;

media[1;6;3;4;5] [1;2;-4;4;6];;
```

Resultado: int list = [1; 4; -1; 4; 5]

1b) Prolog;

```
media([],[],[]).
media([Head1|Rest1],[Head2|Rest2],[ResultHead|RestResult]) :-
    (Head1 >0,Head2 >0)
->  ResultHead is (Head1+Head2)/2, media(Rest1,Rest2,RestResult);
      ResultHead is Head1+Head2, media(Rest1,Rest2,RestResult).

media([1,6,3,4,5],[1,2,-4,4,6],X).
```

Resultado: X = [1, 4, -1, 4, 5.5]

1c) Java;

```
List<Integer> list1 = Arrays.asList(1,6,3,4,5);
List<Integer> list2 = Arrays.asList(1,2,-4,4,6);
List<Integer> listFinal = new ArrayList<Integer>();

for (int i = 0; i < list1.size(); i++) {
	if(list1.get(i)>0 && list2.get(i)>0){
		listFinal.add((list1.get(i)+list2.get(i))/2);
	}else{
		listFinal.add(list1.get(i)+list2.get(i));
	}
}
for (int i = 0; i < listFinal.size(); i++) {
	System.out.print(listFinal.get(i)+" ");
}
```

Resultado: 1 4 -1 4 5

Grupo II

1) O'Caml, diferença entre o valor máximo e o valor mínimo de uma árvore binária.

```
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)));;



let rec top t =

  match t with

    Empty->0

                |Node(v,l,r)-> if (if v > top l then v else top l) > top r
then (if v > top l then v else top l) else top r;;



let rec lower t =

  match t with

    Empty->999999

                |Node(v,l,r)-> if (if v < lower l then v else lower l) <
lower r then (if v < lower l then v else lower l) else lower r;;



top example_tree - lower example_tree;;
```

Resultado:6

2) Prolog**,** resultados positivos Covid numa determinada data.

```
teste_covid(pedro, 20210319, positivo, pcr).
teste_covid(rui, 20210319, negativo, pcr).
teste_covid(pedro, 20210319, positivo, sorologico).
teste_covid(ines, 20210319, negativo, sorologico).
teste_covid(joana, 20210319, positivo, sorologico).
teste_covid(carla, 20210318, positivo, sorologico).

testes_positivos(TipoTeste,Data,Lista) :-
  findall((Utente),teste_covid(Utente,Data,positivo,TipoTeste),Lista).

testes_positivos(sorologico,20210319,Lista).
```

Resultado: Lista = [pedro, joana]

3)

```
import java.util.*;
import java.util.Date;
import java.time.LocalTime


public class Main
{
    public enum Classifica { T, M6, M10, M12, M16, M18 }
    public enum GeneroM { drama, policial, comédia, ação }

    public static class Video {
        private Integer Id;
        private String Titulo
        private LocalTime Duracao;
        private Date Lancamento;
        private Classifica Classificacao;
        private GeneroM Genero;
        private Boolean Disponibilidade;
        private String Sinopse;
        private String Url;
        private List<Video> Videos;
        private List<Visualizacao> Visualizacoes;
        private Integer TotalVideosDisponiveis;

        //Gets
```

```
//....
public List<Video> getVideos() {
        return Videos;
    }
 public List<Visualizacao> getVisualizacoes() {
        return Visualizacao;
 }

//Sets
//....

//Metodos
public void listaVideosDisponiveis(){

    Integer devolve = 0;
    List<Video> videos = getVideos();

    for (int i = 0; i < videos.size(); i++) {
        Video VideoTMP = videos.get(i);
            if(VideoTMP.Disponibilidade){
                devolve ++
            }
    }
    this.TotalVideosDisponiveis = devolve;
}
```

```
    public void ImprimirInformacao(){

        List<Video> videos = getVideos();
        List<Visualizacao> visualizacoes = getVisualizacoes();

        for (int i = 0; i < videos.size(); i++) {
            Integer totalVisualizacoes = 0;
            Integer totalGostos = 0;

            Video VideoTMP = videos.get(i);
                for (int a = 0; a < visualizacoes.size(); a++) {
                    Visualizacao VisualizacaoTMP = visualizacoes.get(a);
                    totalVisualizacoes =
VisualizacaoTMP.totalVisualizacoesPorVideo(VideoTMP.Id);
                    totalGostos =
VisualizacaoTMP.totalGostosPorVideo(VideoTMP.Id);
                }

            System.out.println("Titulo: " + VideoTMP.getTitulo() +
                    "Género,: " + VideoTMP.getGenero()+
                     "Total de visualizações" + totalVisualizacoes +
                     " total de gostos" + totalGostos);
        }

    }
```

```
}
 public static class Visualizacao {

    private Integer IdVideo;
    private String subscritor,
    private Date visualização,;
    private Boolean Gosto;
    private String Sinopse;
    private String Url;
    private List<Visualizacao> Visualizacoes;
    private Integer TotalVisualizacoesPorVideo;
    private Integer TotalGostosPorVideo;

    //Gets
    //....
    public List<Visualizacao> GetVisualizacoes() {
                return Visualizacoes;
        }

    //Sets
    //....

    //Metodos
    public void totalVisualizacoesPorVideo(Integer id){

        Integer devolve = 0;
```

```
            List<Visualizacao> visualizacoes = GetVisualizacoes();


            for (int i = 0; i < visualizacoes.size(); i++) {
                Visualizacao VisualizacaoTMP = visualizacoes.get(i);
                    if(VisualizacaoTMP.IdVideo == id){
                        devolve ++
                    }
            }
            this.TotalVisualizacoesPorVideo = devolve;
        }
        public void totalGostosPorVideo(Integer id){
            Integer devolve = 0;
            List<Visualizacao> visualizacoes = GetVisualizacoes();


            for (int i = 0; i < visualizacoes.size(); i++) {
                Visualizacao VisualizacaoTMP = visualizacoes.get(i);
                    if(VisualizacaoTMP.IdVideo == id &&
VisualizacaoTMP.Gosto){
                        devolve ++
                    }
            }
            this.TotalGostosPorVideo = devolve;
        }
    }
}
```