# *UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO*

## *FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES*

Inteligencia Artificial I

CONTENIDO: Problemas de combinaciones y permutaciones

PORCENTAJE TERMINADO: 100%

GRUPO: 11

| INTEGRANTES | DT | HG | HI | EVAL |
|---|---|---|---|---|
| Flores Veizaga Eudenia Gandira | 1 | 1 | 1 | 100 |
| Garcia Taboada Brayan Albaro | 1 | 1 | 1 | 100 |
| | | | | |

**Fecha de presentación** : Martes, 09 de abril de 2024

**Fecha Presentada :** Martes, 09 de abril de 2024

**Días de Atraso :** 0

```
public class MochilaSubList {

 public static void main(String[] args) {
    LinkedList<Integer> L1= new LinkedList();
    LinkedList<LinkedList<Integer>> L2= new LinkedList();
    L1.add(2);L1.add(5);L1.add(3);L1.add(8);
    System.out.println(L1);
    //mostrarSubListas(L1);
    encontrarSubListas(L1,L2);
    System.out.println(L2);

 }

 public static void mostrarSubListas(LinkedList<Integer> L1){
    for(int i=0;i<L1.size();i++){
       for(int j=i+1;j<=L1.size();j++){
          System.out.println(L1.subList(i, j));
       }
    }
 }

   public static void generaElem(LinkedList<Integer> L1, int n,int a, int b){
      for (int i = 0; i <= n; i++) {
         L1.add((int)(Math.random() * b)+a);
      }
   }


   public static void encontrarSubListas(LinkedList<Integer> L1,
        LinkedList<LinkedList<Integer>> L2){
       for(int i=0;i<L1.size();i++){
```

```
    for(int j=i+1;j<=L1.size();j++){
        L2.add(new LinkedList(L1.subList(i, j)));
    }
 }
}

public static void mostrarIguales(LinkedList<LinkedList<Integer>> L2){
   for(int i=0;i<L2.size();i++){
      if(repetidos(L2.get(i))){
         System.out.println(L2.get(i));
      }

   }
}

private static boolean repetidos(LinkedList<Integer> List) {
   for(int i=0;i<List.size();i++){
      LinkedList<Integer> aux=new LinkedList(List.subList(i+1,List.size()));
      if(aux.contains(List.get(i))){
         return false;
      }
   }
   return true;
}

public static void mostrarDif(LinkedList<LinkedList<Integer>> L2){
   for(int i=0;i<L2.size();i++){
      if(!repetidos(L2.get(i))){
         System.out.println(L2.get(i));
      }
```

```
   }
}

public static void mostrarAsc(LinkedList<LinkedList<Integer>> L2){
   for(int i=0;i<L2.size();i++){
      if(asc(L2.get(i))){
         System.out.println(L2.get(i));
      }
   }
}

 public static boolean asc(LinkedList<Integer> List){
   int may=List.get(0);
    for (int i=1;i<List.size();i++){
         if(List.get(i)>may){
            may=List.get(i);
         }
         else return false;
   }
    return true;

 }

public static void mostrarDesc(LinkedList<LinkedList<Integer>> L2){
   for(int i=0;i<L2.size();i++){
      if(des(L2.get(i))){
         System.out.println(L2.get(i));
      }
   }
}
 public static boolean des(LinkedList<Integer> List){
```

```
   int may=List.get(0);
    for (int i=1;i<List.size();i++){
          if(List.get(i)<may){
             may=List.get(i);
          }
          else return false;
   }
    return true;

 }
public static void mostrarLong(LinkedList<LinkedList<Integer>> L2, int k){
   for (int i = 0; i < L2.size(); i++) {
      if(L2.get(i).size()==k){
         System.out.println(L2.get(i));
      }

   }
}
public static void mostrarPositivos(LinkedList<LinkedList<Integer>> L2){
   for (int i = 0; i < L2.size(); i++) {
      if(positivos(L2.get(i))){
         System.out.println(L2.get(i));
      }

   }
}
public static boolean positivos(LinkedList<Integer> List){
    for (int i=0;i<List.size();i++){
          if(List.get(i)<0){
             return false;
          }
```

```
    }
     return true;

 }

public static void mostrarNegativos(LinkedList<LinkedList<Integer>> L2){
    for (int i = 0; i < L2.size(); i++) {
        if(negativos(L2.get(i))){
            System.out.println(L2.get(i));
        }

    }
}
    public static boolean negativos(LinkedList<Integer> List){
     for (int i=0;i<List.size();i++){
          if(List.get(i)>=0){
              return false;
          }
    }
     return true;

 }

public static void mostrarPosNeg(LinkedList<LinkedList<Integer>> L2){
    for (int i = 0; i <= 10; i++) {

    }
}

public static void mejorSuma(LinkedList<LinkedList<Integer>> L2){
    for (int i = 0; i <= 10; i++) {
```

```
    }
}

public static void peorSuma(LinkedList<LinkedList<Integer>> L2){
    for (int i = 0; i <= 10; i++) {

    }
}
```