II. El problema de la mochila de capacidad MAX

Implementar los siguientes problemas: ingresando los pesos de objetos en una Lista de Enteros L1 y mostrando la combinación de objetos en una Lista de Enteros L2 y utilizando un algoritmo de la forma de llamada recursiva dentro de un ciclo.

Ejercicio 1:

Encontrar todas las combinaciones de pesos de objetos que se pueden transportar en la mochila.

```
public static void mochila(LinkedList<Integer>L1,
                                LinkedList<Integer>L2, int max, int i){
        int sum= suma(L2);
        if(sum>max)return;
        System.out.println(L2);

        int k=i;
        while(k<L1.size()){
            L2.add(L1.get(k));
            mochila(L1,L2,max,k+1);
            L2.removeLast();
            k=k+1;
        }
    }
```

Ejercicio 2:

Encontrar todas las combinaciones de pesos diferentes que se pueden transportar en la mochila.

```
public static void mochilaPesoDif(LinkedList<Integer>L1,
                                LinkedList<Integer>L2, int max, int i){
        int sum= suma(L2);
        eliminarDup(L1);
        if(sum>max)return;
        System.out.println(L2);
        int k=i;
        while(k<L1.size()){
            L2.add(L1.get(k));
            mochila(L1,L2,max,k+1);
            L1.removeLast();
            k=k+1;
        }
    }
```

Ejercicio 3:

Encontrar todas las combinaciones de pesos entre a y b inclusive que se pueden transportar en la mochila.

```
public static void mochila_ab(LinkedList<Integer> L1,LinkedList<Integer> L2,int
    a,int b,int max,int i){
        int sum=suma(L2);
        if(sum>max){return;}
            if(rango(L2,a,b)){
                System.out.println(L2);
            }
        int k=i;
        while(k<L1.size()){
            L2.addLast(L1.get(k));
            mochila_ab(L1,L2,a,b,max,k+1);
            L2.removeLast();
            k++;
        }
    }
```

Ejercicio 4:

Encontrar las combinaciones de objetos de mayor cantidad de objetos que se pueden transportar.

```
public static void mochilaMayorCantidad(LinkedList<Integer> L1,LinkedList<Integer>
    L2,int max,int k){

        LinkedList<Integer> L3 = new LinkedList();
        auxiliar(L1,L2,L3,max,k);
        int may = mayorLista(L3);
        mochilaMayor1(L1,L2,max,k,may);
    }

public static void auxiliar(LinkedList<Integer> L1,LinkedList<Integer>
    L2,LinkedList<Integer> L3,int max,int k){
        int sum = suma(L2);
        if(sum>max)return;
        L3.add(L2.size());
        int i = k;
        while(i<L1.size()){
            L2.add(L1.get(i));
            auxiliar(L1,L2,L3,max,i+1);
            L2.removeLast();
            i++;
        }
    }

public static int mayorLista(LinkedList<Integer> L1){

        int may = 0;
```

```
        for(int i=0;i<L1.size();i++){
            if(L1.get(i)>may){
                may = L1.get(i);
            }
        }
        return may;
    }

 private static void mochilaMayor1(LinkedList<Integer> L1,LinkedList<Integer>
    L2,int max,int k,int cant){
        int sum = suma(L2);
        if(sum>max)return;
        if(L2.size()==cant){
           System.out.println(L2);
        }
        int i = k;
        while(i<L1.size()){
            L2.add(L1.get(i));
            mochilaMayor1(L1,L2,max,i+1,cant);
            L2.removeLast();
            i++;
        }
    }
```

Ejercicio 5:

Encontrar las mejores combinaciones que se pueden transportar en la mochila. (Las más próximas a la capacidad de la mochila)

```
public static void mochilaProx(LinkedList<Integer> L1,LinkedList<Integer> L2,int
    max,int k){

        LinkedList<Integer> L3 = new LinkedList();
        auxiliar2(L1,L2,L3,max,k);
        int prom = promedio(L3);

        mochilaProx1(L1,L2,max,k,prom);
    }

public static void auxiliar2(LinkedList<Integer> L1,LinkedList<Integer>
    L2,LinkedList<Integer> L3,int max,int k){
        int sum = suma(L2);
        if(sum>max)return;
        L3.add(sum);
        int i = k;
        while(i<L1.size()){
```

```
            L2.add(L1.get(i));
            auxiliar2(L1,L2,L3,max,i+1);
            L2.removeLast();
            i++;
        }
    }

public static int promedio(LinkedList<Integer> L1){

        return suma(L1)/L1.size();
    }

    private static void mochilaProx1(LinkedList<Integer> L1,LinkedList<Integer>
       L2,int max,int k,int prom){

        int sum = suma(L2);
        if(sum>max)return;
        if(sum>prom){
            System.out.println(L2);
        }

        int i = k;
        while(i<L1.size()){
            L2.add(L1.get(i));
            mochilaProx1(L1,L2,max,i+1,prom);
            L2.removeLast();
            i++;
        }
    }
```