IV. Propuestos

Ejercicios propuestos de cada punto.

Ejercicio 1:

Sumandos tal que la diferencia de todos los elementos no excede d.

```
public static boolean checkMaximumDiference(LinkedList<Integer> L,int d){
        for(int i=0;i<L.size();++i){
            int u=L.get(i);
            for(int j=i+1;j<L.size();++j){
                int v = L.get(j);
                if(Math.abs(u-v)>d){
                    return false;
                }
            }
        }
        return true;
    }


    public static void sumandosMaxDif(LinkedList<Integer> L,int i,int n,int d){
        int sum = suma(L);
        if(sum>n)return;
        if(sum==n && checkMaximumDiference(L,d)){System.out.println(L);}
        for(int k=i;k<=n;++k){
            L.add(k);
            sumandosMaxDif(L,k,n,d);
            L.removeLast();
        }
    }
```

Ejercicio 2:

Sumandos tal que todos los elementos son numeros pares.

```
public static boolean pares(LinkedList<Integer> L){
        for(int i=0;i<L.size();++i){
            if(L.get(i)%2!=0)
                return false;
        }
        return true;
    }

public static void sumandosPares(LinkedList<Integer> L,int i,int n){
        int sum = suma(L);
        if(sum>n)return;
        if(sum==n && pares(L)){System.out.println(L);}
        for(int k=i;k<=n;++k){
            L.add(k);
            sumandosPares(L,k,n);
            L.removeLast();
        }
    }
```

Ejercicio 3:

Productos donde todos los elementos son impares.

```
public static boolean impares(LinkedList<Integer> L){
        for(int i=0;i<L.size();++i){
            if(L.get(i)%2==0)
                return false;
        }
        return true;
    }


    public static void productosImpares(LinkedList<Integer> L,int i,int n){
        int prod = producto(L);
        if(prod>n)return;
        if(prod==n && impares(L)){System.out.println(L);}
        for(int k=i;k<=n;++k){
            if(n%k==0){
                L.add(k);
                productosImpares(L,k,n);
                L.removeLast();
            }
        }
    }
```

Ejercicio 4:

Productos donde todos los elementos son cuadrados perfectos.

```
public static boolean isPerfectSquare(double x)
    {
        double sr = Math.sqrt(x);
        return ((sr - Math.floor(sr)) == 0);
    }

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

    public static void productosCuadrados(LinkedList<Integer> L,int i,int n){
        int prod = producto(L);
        if(prod>n)return;
        if(prod==n && cuadradosPerfectos(L)){System.out.println(L);}
        for(int k=i;k<=n;++k){
            if(n%k==0){
                L.add(k);
                productosImpares(L,k,n);
                L.removeLast();
            }
        }
    }
```

Ejercicio 5:

Problema 0-1 Knapsack **Referencias:**

https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

https://en.wikipedia.org/wiki/Knapsack_problem

```
static int knapSack(int W,
                                        LinkedList<Integer> wt,
                                        LinkedList<Integer> val, int n)
    {
        // Caso Base
        if (n == 0 || W == 0)
            return 0;

        if (wt.get(n-1) > W)
            return knapSack(W, wt, val, n - 1);
        else
            return Math.max(
                val.get(n-1) + knapSack(W - wt.get(n - 1),
                                        wt, val, n - 1),
                knapSack(W, wt, val, n - 1));
    }
```