Dado un entero N, encontrar todos los factores posibles, enteros positivos de N.

Ejercicio 1:

Encontrar los sumandos posibles en una Lista.

```
public static void factoresa(LinkedList <Integer> L1,int n,int i){
    int mul=mult(L1);
    if(mul>n){return;}
    if(mul==n){
        System.out.println(L1);
        return;
    }
    int k=i;
    while(k<n){
        L1.addLast(k);
        factoresa(L1,n,k);
        L1.removeLast();
        k++;
    }
}
```

Ejercicio 2:

Encontrar todos los sumandos posibles diferentes en una Lista.

```
public static void factoresb(LinkedList <Integer> L1,int n,int i){
    int mul=mult(L1);
    if(mul>n){return;}
    if(mul==n){
        if(diferentes(L1)){
            System.out.println(L1);
        }
        return;
    }
    int k=i;
    while(k<n){
        L1.addLast(k);
        factoresb(L1,n,k);
        L1.removeLast();
        k++;
    }
}
```

Ejercicio 3:

Encontrar todos los sumandos posibles iguales en una Lista.

```
public static void factoresc(LinkedList <Integer> L1,int n,int i){
    int mul=mult(L1);
    if(mul>n){return;}
    if(mul==n){
        if(iguales(L1)){
            System.out.println(L1);
        }
        return;
    }
    int k=i;
    while(k<n){
        L1.addLast(k);
        factoresc(L1,n,k);
        L1.removeLast();
        k++;
    }
}
```

Ejercicio 4:

Encontrar todos los sumandos primos posibles en una Lista.

```
public static void factoresd(LinkedList <Integer> L1,int n,int i){
    int mul=mult(L1);
    if(mul>n){return;}
    if(mul==n){
        if(primos(L1)){
        System.out.println(L1);
        }
        return;
    }
    int k=i;
    while(k<n){
        L1.addLast(k);
        factoresd(L1,n,k);
        L1.removeLast();
        k++;
    }
}
```

Ejercicio 5:

Encontrar todos los sumandos entre a y b inclusive en una Lista.

```
public static void factorese(LinkedList <Integer> L1,int n,int a,int b){
    int mul=mult(L1);
    if(mul>n){return;}
    if(mul==n){
        if(rango(L1,a,b)){
            System.out.println(L1);
        }
        return;
    }
    int k=a;
    while(k<n){
        L1.addLast(k);
        factorese(L1,n,a,b);
        L1.removeLast();
        k++;
    }
}
```

Anexos

Funciones auxiliares para realizar los ejercicios

```
private static int suma(LinkedList<Integer> L1) {
    int sum=0;
    for(Integer j : L1){
        sum=j+sum;
    }
    return sum;
}

private static int mult(LinkedList<Integer> L1) {
    int mul=1;
    for(Integer j: L1){
        mul=mul*j;
    }
    return mul;
}

private static boolean diferentes(LinkedList<Integer> L1) {
    int vec[]=new int[L1.size()];
    int p=-1;
    boolean b=false;
    for(Integer i:L1){
```

```
            p++; vec[p]=i;
        }
        p=0;
        int n=vec.length;
        while(p<n-1 && vec[p]!=vec[p+1]){
            p++;
        }
        if(p==n-1){
            b=true;
        }
        return b;
    }

    private static boolean iguales(LinkedList<Integer> L1) {
        int vec[]=new int[L1.size()];
        int p=-1;
        boolean b=true;
        for(Integer i:L1){
            p++; vec[p]=i;
        }
        p=0;
        int n=vec.length;
        int i=0;
        while(i<n-1 && b==true){
            int j=i+1;
            while(j<n && b==true){
                if(vec[i]!=vec[j]){
                    b=false;
                }
                j++;
            }
            i++;
        }
        return b;
    }

    private static boolean primos(LinkedList<Integer> L1) {
        boolean b=true;
        for(Integer j : L1){
            int x=j;
            if(!pri(x)){
                b=false;
            }
        }
        return b;
```

```
}

private static boolean pri(int x) {
    boolean b=true;
    for(int mu=2;mu<x;mu++){
        if(x%mu==0){
            b=false;
        }
    }
    return b;
}

private static boolean rango(LinkedList<Integer> L1,int a,int b) {
    boolean boo=true;
    for(Integer j : L1){
        int x=j;
        if(x<a || x>b){
            boo=false;
        }
    }
    return boo;
}
```