Problema 20

Dada una lista L de n elementos $a_1, a_2, ..., a_n$ y un íntervalo $[b, c]$ dentro de esa lista devolver el máximo elemento de ese intervalo.

- b : un número de tamaño db que representa b
- c : un número de tamaño db que representa c
- n : un número de tamaño dd que representa al tamaño de la lista L
- $array$: un array de números de tamaño dd que representa L

```
Solve 20(b, c, n, L):
    count = b
    max = [L + count]

    while b <= c:
        if max < [L + count]
            max = [L + count]
        count++

    return max
```

Problema 45

Sea una lista no ordenada L de n elementos $a_1, a_2, ..., a_n$ y un número x, devolver la suma de todos los números en L que sean mayores que x.

- x : un número de tamaño db que representa x
- n : un número de tamaño dd que representa al tamaño de la lista L
- $array$: un array de números de tamaño dd que representa L

```
Solve 45(x, n, L):
    sum  = 0
    count = 0

    while count < n:
        if [L + count] > x
            sum += [L + count]
        count++

    return sum
```

Problema 66

Dado dos arrays ordenados A_1, A_2 devolver el arreglo ordenado A_3 obtenido haciendo una mezcla ordenada de A_1 y A_2.

- n_1 : un número de tamaño dd que representa al tamaño de la lista L
- $array_1$: un array de números de tamaño dd que representa L
- n_2 : un número de tamaño dd que representa al tamaño de la lista $L^{'}$
- $array_2$: un array de números de tamaño dd que representa $L^{'}$

```
Solve 66(n1, L1, n2, L2):
    sort = new array[n1 + n2]
    count = 0
    count1 = 0
    count2 = 0

    while count < n1 + n2:

        if count1 == n1:
            goto Incrementar2

        if count2 == n2:
            goto Incrementar1

        if [L1 + count1] < [L2 + count2]:
            goto Incrementar1
        else:
            goto Incrementar2

        Incrementar1:
        [sort + count] = [L1 + count1]
        count++
        count1++
        continue

        Incrementar2:
        [sort + count] = [L2 + count2]
        count++
        count2++
        continue

    return sort
```

Problema 81

Se tiene una lista L de tamaño n con números enteros. Construya la lista L' tal que

$$L'[i] = \max_{j=0}^{i} L[j]$$

con $i = 0, 1, ..., n-1$.

- n : un número de tamaño dw que representa n
- L : un array de números de tamaño dw que representa L

```
Solve 81(n, L):
    maximum = new arr[n]
    count = 0
    max = [L]

    while count < n:
        if max < [L + count]
            max = [L + count]

        [maximum + count] = max
        count++

    return maximum
```