**ADA – Tarea 2**
**Xavier Garzón**

**Ejercicio 2:** ***Computer hackers***

***a)***

```
For iterations i = 1 to n
  If h_{i+1} > ℓ_i + ℓ_{i+1} then
    Output "Choose no job in week i"
    Output "Choose a high-stress job in week i + 1"
    Continue with iteration i + 2
  Else
    Output "Choose a low-stress job in week i"
    Continue with iteration i + 1
  Endif
End
```

| | Week 1 | Week 2 | Week 3 | Week 4 |
|---|---|---|---|---|
| ***l*** | 10 | 1 | 10 | 10 |
| ***h*** | 5 | 50 | 100 | 1 |

$$i = 1$$

$$h_{i+1} > l_i + l_{i+1}$$
$$h_2 > l_1 + l_2$$
$$50 > 10 + 1$$
$$50 > 11$$
$$ans_1 = 50$$

$$i = 3$$

$$h_{i+1} > l_i + l_{i+}$$
$$h_4 > l_3 + l_4$$
$$1 > 10 + 1$$
$$1 > 20$$
$$ans_2 = 10$$

$$i = 4$$

$$h_{i+1} > l_i + l_{i+1}$$
$$h_5 > l_4 + l_5$$
$$0 > 10 + 0$$
$$0 > 10$$
$$ans_3 = 10$$

$$ans_1 + ans_2 + ans_3 = 70$$

En este ejemplo, la respuesta correcta sería 120. Pero con haciendo una simple prueba podemos notar que el algoritmo dado nos da como respuesta 70.

***b)***

```python
import math
INF=math.inf
def hackers(n,L,H):
    """
    Este algoritmo tiene como objetivo calcular el
    máximo beneficio que puede obtener un grupo de
    trabajo en una cantidad de semanas dadas y unos
    tipos de trabajos pagados según la semana en
    que se realicen.

    Entrada:
        n -> número de semanas
        L -> arreglo con las ganancias estrés bajo
        H -> arreglo con las ganancias estrés alto
    Salida: máximo beneficio con un plan según las
            entrada
    """
    global DATOS #arreglo para memorizar

    if DATOS[n]!=-INF:
        ans=DATOS[n]
    else:
        if n<0:
            ans=0
        else:
            ans=max(hackers(n-1,L,H)+L[n-1],
                    hackers(n-1,L,H)+0,
                    hackers(n-2,L,H)+H[n-1])
        DATOS[n]=ans
    return ans
#Complejidad temporal: O(n)
```

**Ejercicio 6:** ***String shuffling***

***a)***

```
1  def bananas(A,B,C):
2      """
3      Este algoritmo tiene como objetivo
4      verificar si dos cadenas dadas están
5      mezcladas en una tercera cadena.
6
7      Entrada:
8          A -> arreglo con la palabra A
9          B -> arreglo con la palabra B
10         C -> arreglo con la posible
11             mezcla de A y B
12     Salida: True si A y B están en C
13             False de otro modo
14     """
15     a=0; b=0
16     ans=True
17     if len(A)+len(B)!=len(C):
18         ans=False
19     else:
20         while ans and (a<len(A) and b<len(B)):
21             if A[a]==C[a]:
22                 a+=1
23             elif B[a]==C[a]:
24                 b+=1
25             else:
26                 ans=False
27     return ans
28 #Complejidad temporal:
```