

DOCUMENTO TÉCNICO

Requisitos

Especificación1

Escriba una función recursiva que ordene de menor a mayor una lista de enteros basándose en la siguiente idea: coloque el elemento más pequeño en la primera ubicación, y luego ordene el resto del arreglo con una llamada recursiva.

Entrada

En la primera línea se digita la cantidad de números que se desea y en la segunda línea cada número de acuerdo a la cantidad puesta anteriormente, separado por espacio entre cada número. Se toman números enteros, pero la cantidad de números mayor a 0.

Salida

Se generan los números organizados ascendentemente en forma de lista.

Especificación2

Escribir una función recursiva que devuelva la suma de un subarreglo de N enteros, limitado por índices (i, j) en una lista de enteros L.

Entrada

En la primera línea se digita la cantidad de números que se desea, en la segunda línea cada número de acuerdo a la cantidad puesta anteriormente, separado por espacio entre cada número; y en la tercera línea se digitan el intervalo que se desea, también separado por espacio. Se toman números enteros, pero la cantidad de números mayor a 0.

Salida

Se genera la suma de los números que están en el intervalo.

Especificación3

Escribir una función y un programa que encuentre la suma de los enteros positivos pares desde N hasta 2.

Entrada

En la única línea se digita el número que sea entero positivo.

Salida

Se genera la suma de los números enteros positivos desde el número ingresado hasta 2.

Especificación4

Dada una función recursiva para MCD cómo

$MCD = M$ si $N = 0$

$MCD = MCD(N, M \bmod N)$ si $N \neq 0$

Escriba un programa que le permita al usuario ingresar los valores para M y N desde la consola. Una función recursiva es entonces llamada para calcular el MCD. El programa entonces imprime el valor para el MCD.

Entrada

En la primera línea se digitan los dos números a los cuales se va a calcular el MCD, separado por espacio.

Salida

Se muestra el máximo común divisor de los dos números.

Especificación5

Programa un método recursivo que transforme un número entero positivo a notación binaria.

Entrada

Se digita el número entero positivo que se quiere mostrar en binario en la primera línea.

Salida

Se muestra el número en notación binaria.

Especificación6

Programa un método recursivo que invierta los números de un arreglo de enteros.

Entrada

En la primera línea se digita la cantidad de números que se desea y en la segunda línea cada número de acuerdo a la cantidad puesta anteriormente, separado por espacio entre cada número. Se toman números enteros, pero la cantidad de números mayor a 0.

Salida

Se generan el arreglo invertido de los números en forma de lista.

Diseño

Estrategia1

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por pila, porque va desde encontrar el mínimo del arreglo y añadir lo que sobra, y así hasta que el arreglo tenga 1 elemento.

Handwritten recursive function for sorting an array:

$$\text{Función recursiva}$$
$$A) \text{ Ordenar}(A) = \begin{cases} A & |A| \leq 1 \\ [\min(A)] + \text{Ordenar}(A - [\min(A)]) & |A| > 1 \end{cases}$$

La estructura de datos a implementar será la lista, y se utilizará para guardar los inputs del usuario, y también para almacenar los números y la salida organizada ascendentemente.

Estrategia2

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por pila, porque va desde encontrar el primer intervalo del arreglo definido en la entrada hasta que llegue al intervalo final.

Handwritten recursive function for summing an array:

$$B) \text{ Sumatoria}(i, j) = \begin{cases} \emptyset & i > j \vee i > |A| \\ A[i] + \text{Sumatoria}(i+1, j) & i \leq j \end{cases}$$

La estructura de datos a implementar será la lista, y se utilizará para guardar los inputs del usuario, y también para almacenar los números.

Estrategia3

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por cola, porque compara el número y va recorriendo en términos de posiciones anteriores.

$$c) \text{ SumaPares}(n) = \begin{cases} \emptyset & n \leq 2 \\ \text{SumaPares}(n-1) & n \bmod 2 \neq 0 \\ n + \text{SumaPares}(n-2) & n \bmod 2 = 0 \end{cases}$$

Estrategia4

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por cola, porque compara los dos números mediante un operador y va recorriendo en términos de posiciones anteriores.

$$d) \text{ MCD}(N, M) = \begin{cases} N & M = 0 \\ \text{MCD}(M, N \bmod M) & M \neq 0 \end{cases}$$

Estrategia5

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por cola, porque compara el número cumpliendo la condición en términos anteriores para mostrar el número en notación binaria.

$$e) \text{ Binary}(n) = \begin{cases} "0" & n = 0 \\ "1" & n = 1 \\ \text{Binary}(n \div 2) + \text{str}(n \bmod 2) & n > 1 \end{cases}$$

Da la salida como si la notación binaria fuera un string.

Estrategia6

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por cola, porque empieza tomando la última posición del arreglo y luego agrega lo que falta.

$$f) \text{ReverseNumber}(A) = \begin{cases} [A] & , |A|=0 \\ A[-1] + \text{ReverseNumber}(A[: -1]) & , |A|>0 \end{cases}$$

La estructura de datos a implementar será la lista, y se utilizará para guardar los inputs del usuario, y también para almacenar los números y la salida revertida.

Casos de prueba1

Entrada	Justificación	Salida
4 6 2 8 -7	Cualquier arreglo	[-7,2,6,8]
1 -5	Longitud 1	[-5]
5 0 42 -1 817 -41	Otro arreglo	[-41,-1,0,42,817]

entrada.txt	salida.txt
1 1	1 [-5]
2 -5	2 [-7, 2, 6, 8]
3 4	3 [-41, -1, 0, 42, 817]
4 6 2 8 -7	4
5 5	
6 0 42 -1 817 -41	

Casos de prueba2

Entrada	Justificación	Salida
3 0 5 -7 1 2	Cualquier arreglo	-2
1 4 1 3	Longitud 1	0
5 -5 7 3 0 -1 2 4	Otro arreglo	2

The screenshot shows a code editor with three files: `entrada2.txt`, `SUMA_ARRAY_IN.py`, and `salida2.txt`. The `entrada2.txt` file contains the following input:

```
1 3
2 0 5 -7
3 1 2
4 1
5 4
6 1 3
7 5
8 -5 7 3 0 -1
9 2 4
```

The `salida2.txt` file contains the following output:

```
1 La suma de los elementos entre los índices 1 y 2 es: -2
2 La suma de los elementos entre los índices 1 y 3 es: 0
3 La suma de los elementos entre los índices 2 y 4 es: 2
4
```

Casos de prueba3

Entrada	Justificación	Salida
5	Cualquier número	6
1	Caso restrictivo	0
45	Otro número	56

The screenshot shows a code editor with three files: `entrada3.txt`, `SUMAPARES.py`, and `salida3.txt`. The `entrada3.txt` file contains the following input:

```
1 5
2 1
3 14
```

The `salida3.txt` file contains the following output:

```
1 La suma de los pares desde 5 hasta 2 es de 6
2 La suma de los pares desde 1 hasta 2 es de 0
3 La suma de los pares desde 14 hasta 2 es de 56
4
```

Casos de prueba4

Entrada	Justificación	Salida
18 24	Cualesquiera números	6
5 0	Caso restrictivo	5
30 45	Otros números	15

entrada4.txt	MCD.py	salida4.txt
1 18 24	1 El Máximo Común Divisor (MCD) de 18 y 24 es 6	
2 5 0	2 El Máximo Común Divisor (MCD) de 5 y 0 es 5	
3 30 45	3 El Máximo Común Divisor (MCD) de 30 y 45 es 15	
	4	

Casos de prueba5

Entrada	Justificación	Salida
5	Cualquier número	101
0	Caso restrictivo	0
123	Otro número	1111011

entrada5.txt	BINARY_NUMBER.py	salida5.txt
1 5	1 La representación binaria de 5 es 101	
2 0	2 La representación binaria de 0 es 0	
3 123	3 La representación binaria de 123 es 1111011	
	4	

Casos de prueba6

Entrada	Justificación	Salida
6 -4 7 8 1 -3 43	Cualquier arreglo	[43,-3,1,8,7,-4]
1 1	Longitud arreglo 1	[1]
2 -6 5	Otro arreglo	[5,-6]

entrada6.txt	REVERSE_NUMBERS.py	salida6.txt
1 6	1 Arreglo invertido: [43, -3, 1, 8, 7, -4]	
2 -4 7 8 1 -3 43	2 Arreglo invertido: [1]	
3 1	3 Arreglo invertido: [5, -6]	
4 1	4	
5 2		
6 -6 5		

Análisis

Temporal1

	#Cost (0)	#Times (0)	#Cost (0hm)	#Times (0hm)
23 <code>def menor_mayor(lista):</code>				
24 <code>if len(lista) <= 1:</code>	#c1	n	c1	n
25 <code>return lista</code>	#c2	1	c2	1
26 <code>else:</code>				
27 <code>menorNumero = min(lista)</code>	#c3	1	c3	1
28 <code>lista.remove(menorNumero)</code>	#c4	n-1	c4	n
29				
30 <code>return [menorNumero] + menor_mayor(lista)</code>	#c5	1	c5	1
31	#T(n)=0(n)		#T(n)=0hm(n)	

Temporal2

	#Cost (0)	#Times (0)	#Cost (0hm)	#Times (0hm)
30 <code>def sumaElementos(lista, i, j):</code>				
31 <code>if i > j or i >= len(lista):</code>	#c1	n	c1	n
32 <code>return 0</code>	#c2	1	c2	1
33 <code>else:</code>				
34 <code>return lista[i] + sumaElementos(lista, i+1, j)</code>	#c3	n	c3	n-1
35	#T(n)=0(n)		#T(n)=0hm(n)	

Temporal3

	#Cost (0)	#Times (0)	#Cost (0hm)	#Times (0hm)
5 <code>def sumaPares(n):</code>				
6 <code>if n < 2:</code>	#c1	n	c1	n
7 <code>return 0</code>	#c2	1	c2	1
8 <code>elif n % 2 != 0:</code>	#c3	n	c3	n
9 <code>return sumaPares(n-1)</code>	#c4	n-1	c4	n-1
10 <code>else:</code>				
11 <code>return n + sumaPares(n-2)</code>	#c5	1	c5	1
12	#T(n)= 0(n)		#T(n)= 0hm(n)	

Temporal4

	#Cost (0)	#Times (0)	#Cost (0hm)	#Times (0hm)
11 <code>def mcd(M, N):</code>				
12 <code>if N == 0:</code>	#c1	N	c1	N
13 <code>return M</code>	#c2	1	c2	1
14 <code>else:</code>				
15 <code>return mcd(N, M % N)</code>	#c3	1	c3	1
16	#T(n)= 0(N)		#T(n)= 0hm(N)	

Temporal5

5	def a_binario(n):	#Cost (0)	#Times (0)	#Cost (0hm)	#Times (0hm)
6	if n == 0:	#c1	n	c1	n
7	return '0'	#c2	1	c2	1
8	elif n == 1:	#c3	n	c3	n
9	return '1'	#c4	1	c4	1
10	else:				
11	return a_binario(n // 2) + str(n % 2)	#c5	n	c5	n/2
12			# T(n)= 0(n)		#T(n)= 0hm(n)
	1 usage new *				

Temporal6

	2 usages	new *																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
--	----------	-------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Código

Documentación1

```
1 """Escriba una función recursiva que ordene de menor a mayor una lista de enteros
2 basándose en la siguiente idea: coloque el elemento más pequeño en la primera
3 ubicación, y luego ordene el resto del arreglo con una llamada recursiva. """
4
5 """Librería para que los inputs se pongan en un archivo"""
6 import sys
7
8 new *
9
10 def carga_numeros():
11     lista = []
12     n = int(sys.stdin.readline().strip())
13     numeros = sys.stdin.readline().strip().split()
14
15     #carga_numeros administra los datos(cantidad y números)
16
17     for i in range(n):
18         try:
19             numero = int(numeros[i])
20         except ValueError:
21             continue
22
23     lista.append(numero)
24
25     return lista
```

```

23 def menor_mayor(lista):                                #Cost (0)   #Times (0)   #Cost (0hm)   #Times (0hm)
24     if len(lista) <= 1:                                #c1         n         c1         n
25         return lista                                  #c2         1         c2         1
26     else:
27         menorNumero = min(lista)                       #c3         1         c3         1
28         lista.remove(menorNumero)                     #c4         n-1       c4         n
29
30     return [menorNumero] + menor_mayor(lista)          #c5         1         c5         1
31
32                                     #T(n)=O(n)       #T(n)=O(n)

```

```

32 def main():
33     with open('entrada.txt', 'r') as f: # Abre el archivo 'pruebas.txt' en modo lectura
34         while True:
35             try:
36                 line = next(f).strip() # Lee la siguiente línea y elimina los espacios en blanco
37                 if not line: # Si la línea está vacía, pasa a la siguiente línea
38                     continue
39                 n = int(line) # Convierte la línea en un número entero
40                 numeros = next(f).strip().split() # Lee los números de la prueba
41                 lista = [int(num) for num in numeros]
42                 listaAscendente = menor_mayor(lista)
43                 print(listaAscendente)
44             except StopIteration: # Cuando no hay más pruebas, termina el bucle
45                 break
46 main()
47

```

Documentación2

```

1  """Escribir una función recursiva que devuelva la suma de un subarreglo de N enteros,
2  limitado por índices (i,j) en una lista de enteros L"""
3
4  import sys                                #Librería
5
6  new *
7  def número_Elementos():
8      n=-1
9
10     while n <= 0:
11         try:
12             n=int(sys.stdin.readline().strip()) #número_Elementos administra la cantidad de números
13         except ValueError:
14             continue
15     return n

```

```

16 def carga_numeros(n):
17     lista = []
18     numeros = sys.stdin.readline().strip().split()
19
20     for i in range(n):
21         try:                                #carga_numeros administra los números
22             numero = int(numeros[i])
23         except ValueError:
24             continue
25
26     lista.append(numero)
27
28     return lista
29

```

```

30 def sumaElementos(lista, i, j):
31     if i > j or i >= len(lista):
32         return 0
33     else:
34         return lista[i] + sumaElementos(lista, i+1, j)
35
1 usage new *
#Cost (0) #Times (0) #Cost (0hm) #Times (0hm)
#c1 n c1 n
#c2 1 c2 1
#c3 n c3 n-1
# T(n)=0(n) #T(n)=0hm(n)

36 def main():
37     with open('entrada2.txt', 'r') as f: # Abre el archivo 'entradas.txt' en modo lectura
38         while True:
39             try:
40                 n = int(next(f)) # Lee la cantidad de números en la prueba
41                 numeros = next(f).strip().split() # Lee los números de la prueba
42                 lista = [int(num) for num in numeros]
43                 i, j = map(int, next(f).strip().split()) # Lee los índices i y j
44                 suma = sumaElementos(lista, i, j)
45                 print("La suma de los elementos entre los índices", i, "y", j, "es:", suma)
46             except StopIteration: # Cuando no hay más pruebas, termina el bucle
47                 break
48 main()
49

```

Documentación3

```

1 """Escribir una función y un programa que encuentre la suma de los enteros positivos pares desde N hasta 2."""
2
3 import sys #Libreria
4
5 3 usages new *
6 def sumaPares(n):
7     if n < 2:
8         return 0
9     elif n % 2 != 0:
10         return sumaPares(n-1)
11     else:
12         return n + sumaPares(n-2)
13
14 1 usage new *
15 def main():
16     for line in sys.stdin:
17         try:
18             n = int(line.strip())
19             sumaParesF = sumaPares(n)
20             print("La suma de los pares desde", n, "hasta 2 es de", sumaParesF)
21
22 main()
23
#Cost (0) #Times (0) #Cost (0hm) #Times (0hm)
#c1 n c1 n
#c2 1 c2 1
#c3 n c3 n
#c4 n-1 c4 n-1
#c5 1 c5 1
#T(n)= 0(n) #T(n)= 0hm(n)
#Número a tener en cuenta

```

Documentación4

```

1  """Dada una función recursiva para MCD cómo
2
3  MCD = M si N = 0
4  MCD = MCD (N, M mod N) si N <> 0
5
6  Escriba un programa que le permita al usuario ingresar los valores para M y N desde la consola.
7  Una función recursiva es entonces llamada para calcular el MCD. El programa entonces imprime el valor para el MCD."""
8
9  import sys          #Libreria
10
11  2 usages new *
12  def mcd(M, N):
13      #Cost (0)      #Times (0)      #Cost (Ohm)      #Times (Ohm)
14      if N == 0:
15          #c1          N          c1          N
16          return M
17          #c2          1          c2          1
18      else:
19          return mcd(N, M % N)
20          #c3          1          c3          1
21          # T(n)= 0(N)          #T(n)= 0hm(N)
22
23  1 usage new *
24  def main():
25      for line in sys.stdin:
26          try:
27              M, N = map(int, line.strip().split())
28              #Los números para calcular MCD
29          except ValueError:
30              continue
31          result = mcd(M, N)
32          print("El Máximo Común Divisor (MCD) de", M, "y", N, "es", result)
33
34  main()

```

Documentación5

```

1  "Programa un método recursivo que transforme un número entero positivo a notación binaria."
2
3  import sys          #Libreria
4
5  2 usages new *
6  def a_binario(n):
7      #Cost (0)      #Times (0)      #Cost (Ohm)      #Times (Ohm)
8      if n == 0:
9          #c1          n          c1          n
10         return '0'
11         #c2          1          c2          1
12     elif n == 1:
13         #c3          n          c3          n
14         return '1'
15         #c4          1          c4          1
16     else:
17         return a_binario(n // 2) + str(n % 2)
18         #c5          n          c5          n/2
19         # T(n)= 0(n)          #T(n)= 0hm(n)
20
21  1 usage new *
22  def main():
23      for line in sys.stdin:
24          try:
25              n = int(line.strip())
26              #Número para pasar a notación binaria
27          except ValueError:
28              continue
29          resultado = a_binario(n)
30          print("La representación binaria de", n, "es", resultado)
31
32  main()

```

Documentación6

```
1 "Programa un método recursivo que invierta los números de un arreglo de enteros"
2
3 import sys #Libreria
4
5 new *
6 def lista():
7     lista = []
8     n = int(sys.stdin.readline().strip()) #lista tiene cantidad y los números del arreglo
9     numeros = sys.stdin.readline().strip().split()
10
11     for i in range(n):
12         try:
13             numero = int(numeros[i])
14         except ValueError:
15             continue
16
17         lista.append(numero)
18
19     return lista
```

	2 usages	new *					
20	def reverse_Array(list):		#Cost (0)	#Times (0)	#Cost (0hm)	#Times (0hm)	
21	if len(list) == 0:		#c1	n	c1	n	
22	return []		#c2	1	c2	1	
23	else:						
24	return [list[-1]] + reverse_Array(list[:-1])		#c3	n-1	c3	n-1	
25			# T(n)= 0(n)		#T(n)= 0hm(n)		
26							

```

1 usage new*
27 def main():
28     with open('entrada6.txt', 'r') as f: # Abre el archivo 'entradas.txt' en modo lectura
29         while True:
30             try:
31                 line = next(f).strip() # Lee la siguiente línea y elimina los espacios en blanco
32                 if not line: # Si la línea está vacía, pasa a la siguiente línea
33                     continue
34                 n = int(line) # Convierte la línea en un número entero
35                 numeros = next(f).strip().split() # Lee los números de la prueba
36                 lista = [int(num) for num in numeros]
37                 ArregloInvertido = reverse_Array(lista)
38                 print("Arreglo invertido: ", ArregloInvertido)
39             except StopIteration: # Cuando no hay más pruebas, termina el bucle
40                 break
41
42 main()

```

Fuentes

/ MENOR-MAYOR.py	/entrada.txt	/salida.txt
/ SUMA_ARRAY_IN.py	/entrada2.txt	/salida2.txt
/ SUMAPARES.py	/entrada3.txt	/salida3.txt
/ MCD.py	/entrada4.txt	/salida4.txt
/ BINARY_NUMBER.py	/entrada5.txt	/salida5.txt
/ REVERSE_NUMBERS.py	/entrada6.txt	/salida6.txt