RETO 2: Abstracción

Participantes:
Rubén Mogica Garrido
Arturo Cortes Sánchez

Para resolver este reto hemos dividido el problema en dos partes. La primera parte es una función recursiva que hemos denominado CyL, la segunda es una función que aborda su parte mediante fuerza bruta.

CyL:

La función CyL recibe:

- un vector ordenado de enteros que contiene los números con los que podemos operar
- un entero el cual es el que tenemos que alcanzar
- un vector vacío en el que se van a almacenar los números del primer vector que no han sido usados

Además hace uso de un vector global donde guarda los resultados

En la función recursiva se realiza una división entera del numero recibido entre el ultimo elemento del vector, y se almacena el cociente y el resto. Si el ultimo elemento del vector es mayor al numero entonces el elemento se guarda en el vector vacío para uso posterior. En caso contrario añade el dividendo, divisor, cociente y resto al vector de resultados. Para evitar repeticiones y que la recursión tenga fin, eliminamos el ultimo elemento del vector de números con los que operar. Finalmente si el cociente no está en la lista de números operables, lanzamos recursivamente la función con el cociente como entero a alcanzar y el vector con los números restantes como posibles operandos y hacemos lo mismo para el resto.

Bruteforce:

Recibe:

- Un vector de números no usados anteriormente con los que operar
- el numero a alcanzar
- el resultado obtenido por la anterior función

Comprueba si el numero es mas grande que el resultado, si es así va sumando los numeros del vector hasta obtener un numero que sumado al resultado supere al numero original. En caso de ser menor, va restando los números para obtener un numero negativo

El programa primero lanza la función CyL con los valores que se desean usar, cuando dicha función acaba, se trata de reconstruir el numero original con un bucle y el vector de resultados. Una vez obtenida esta solución primaria se llama a bruteforce, el cual generará un numero. Si la solución primaria mas dicho numero se acerca mas al numero original, entonces la la solución final será la suma de dichos números. En caso contrario la solución primaria se mantiene como solución final

```
import random

def CyL(val, n, unused, op):
      if(len(val)>0):
      c=n // val[-1]
      r=n % val[-1]
      print(n,"=",val[-1],"*",c,"+",r)

      if n < val[-1]:
            print(val[-1]," sin usar")
            unused.append(val[-1])
      else:
            op.append([n,val[-1],c,r])
      val.pop(-1)
      if c not in val and c!=0 and c!=1:
            CyL(val,c,unused, op)
      if r not in val and r!=0 and r!=1:
            CyL(val,r,unused, op)

def bruteforce(val, num, res):
      i=0
      out=0
      if num > res:
            while i < len(val):
      if num > res+out:
            out+=val[i]
      i+=1
      elif num < res:
            while i < len(val):
      if num < res+out:
            out-=val[i]
      i+=1

return out


vals=sorted(random.sample([1,2,3,4,5,6,7,8,9,10,50,75,100], 6))
vals_backup=vals.copy()

operaciones=[]
unused=[]
num=random.randint(100,1000)
print("num: ",num)
print(vals)

if num==100:
      print("encontrado 100 usando 100")
```

```
else:
      CyL(vals, num, unused, operaciones)

unused=sorted(list(set(vals)|set(unused)))
print("unused ",unused)

print("valores iniciales",vals_backup)

r_op=list(reversed(operaciones))
res=0
i=0
sumandos=[]

while i<len(r_op):
      if r_op[i][2] != 1:
            j=min(vals_backup, key=lambda x:abs(x-r_op[i][2]))
            sumandos.append(r_op[i][1]* j)
      else:
            sumandos.append(r_op[i][1])
      i+=1

for s in sumandos:
      res+=s

print("resultado de CyL",res)
print("sumandos ", sumandos)

extra=bruteforce(unused, num, res)

print("extra ",extra)

if abs(num-res) < abs(num-(res+extra)):
print("resultado definitivo 1", res)
else:
print("resultado definitivo 2", res+extra)

print("operaciones ", operaciones)
```