**Taller Lab2 recursividad**

**Juan Camilo Rojas Castro**

**(En cada archivo esta el print de los test’s)**

**1.** Función de ordenamiento recursivo:

El caso limite es cuando la lista tiene uno o cero elementos en tal caso se devuelve el elemento o ningún elemento.

En el resto de casos escogemos el menor elemento y lo quitamos del arreglo original, este lo metemos en uno nuevo y volvemos a llamar a la función, esto hasta que la lista principal sea como el caso base.

En cuanto al invariante dependerá de la longitud de la lista, para la inicialización la longitud es diferente de cero o uno, en la estabilidad esta sigue siendo mayor a uno, para la longitud es uno y por tanto solo queda un elemento.

Código:

def sortRec(A):  
 if len(A) <= 1:  
 return A  
 minValue = min(A)  
 A.remove(minValue)  
 return [minValue] + sortRec(A)  
  
def main():  
 print(sortRec([2,6,7,4,5,1,8,9]))  
  
main()

**2.** Función suma de los primero N enteros de un arreglo:

El caso limite es cuando la cantidad de elementos N que se va a sumar es inexistente.

En el resto de casos se va a ir sumando los elementos desde N-1 hasta el elemento 0.

El invariante depende de la posición de N, si esta es menor a 0 no se toma en cuenta las posiciones negativas en un arreglo y no hay suma, si N > a 0 se inicializará, en la estabilidad se irán sumando los elementos en las posiciones de N mientras este va decreciendo en 1, para la finalización N cera menor a 0 y se terminara la suma.

Código:

def sumRec(A, N):  
 return 0 if N < 1 else A[N] + sumRec(A, N-1)  
  
def main():  
 print(sumRec([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 4))  
  
main()

**3.** Función suma de los primeros N pares enteros:

El caso limite es cuando ci es mayor a N.

En el resto de casos se suma desde un caso inicial, normalmente 0 hasta N, al hacer la entrada recursiva con (ci+2) se sumarán solo elementos pares.

Nuestro invariante depende del valor de ci, pues en este se inicializa con un ci menor a N, en la estabilidad va a ir sumando par más par hasta llegar a N, para finalizar ci va a pasar a N.

Código:

def sumPairs (ci, N):  
 return 0 if ci > N else ci + sumPairs(ci+2, N)  
  
  
def main():  
 print(sumPairs(0, 11))  
  
  
main()

**4.** Función MCD:

En el caso limite N es 0 por tanto M es el MCD.

En el resto de casos se va a escoger el mayor entre M y N y se va a hacer el mod entre M y N hasta que N sea 0.

El invariante depende del valor de N pues si este es 0 el MCD es M, para inicializar ambos deben ser diferentes de cero, en la estabilidad a M se le hace mod N y el nuevo M es N, para finalizar el N debe ser 0 y M es el MCD.

Código:

def MCD(M, N):  
 N, M = min(M, N), max(M, N)  
 print(M, N)  
 return M if N == 0 else MCD(N, M % N)  
  
def main():  
 print(MCD(15, 6))  
  
main()

**5.** Función que transforma un numero decimal a binario:

El caso limite es cuando n es menor a 1 porque no está definida la división entera entre 2.

Para el resto de casos se hace el modulo entre 2 para obtener el valor de n en 0 o 1 y seguir dividiendo entre dos hasta alcanzar el caso limite.

El invariante esta determinado por el valor de n, para inicializar este debe ser mayor a 1, en la estabilidad n se divide entre dos, para finalizar este es menor o igual a uno.

Código:

def binar(n):  
 return None if n < 1 else binar(n//2) + [n % 2]  
  
def main(n):  
 return binar(n)

**6.** Función que invierte un arreglo de enteros:

El caso limite es cuando la longitud del arreglo es 1 pues en este caso se devuelve el mismo arreglo.

En los otros casos se quita el ultimo elemento de la lista y se agrega a una nueva.

El invariante va a depender de la longitud del arreglo pues para inicializar este debe ser mayor a uno, en la estabilidad este va a ir disminuyendo en 1 hasta la finalización donde va a ser el caso limite.

Código:

def inverseRec(A):  
 if len(A) == 1:  
 return A  
 i = A[len(A)-1]  
 A.remove(i)  
 return [i] + inverseRec(A)  
  
  
def main(A):  
 return inverseRec(A)