Parte B: Introducción a los lenguajes dinámicos

Integrantes de la resolución:	***Borquez, Juan Manuel***	***13567***
	Dalessandro, Francisco	***13318***
	Panonto, Valentín	***13583***
	Welch, Patricio	***13612***
	Zamora, Braulio	***13571***

Link al replit con el código de cada ejercicio:
https://replit.com/@ValentinPanonto/TP4ParteB#main.py

EJERCICIO 5

Implemente el ejercicio 2 (Batalla naval espacial) siguiendo el análisis algorítmico realizado oportunamente. Además responda, mediante comentarios en el código, las siguientes preguntas:

(a) ¿Qué estructura de datos utilizó para representar el tablero del juego, considerando que python no soporta los arreglos nativamente? Justifique.

(b) Sirvió hacer un análisis y un diseño algorítmico previo a la implementación del programa? Justifique.

(c) ¿Hubo elementos (entradas, salidas, requerimientos, etc) que no se habían tomado en cuenta durante su análisis y que aparecieron en el desarrollo? ¿Cuáles?

(d) ¿La separación lógica del problema se vio plasmada en las funciones implementadas en el código?

"""

RESPUESTAS A LAS PREGUNTAS DEL ENUNCIADO

(a)

Para representar el tablero se utilizó un diccionario con las filas (A, B, C, D) como clave y listas de 4 elementos como contenido (cada una de las filas). De esta manera se puede acceder a los elementos del tablero con la clave de la fila como primer índice y con la posición en la lista como índice de columna

(b)

Si, porque permitió tener una idea general de la lógica del programa, de las variables que serían necesarias definir, las salidas que serían importantes para el usuario, etc. Además la definición de funciones en el diseño algorítmico permitió hacer más rápidamente la modularización del programa

(c)

Si

Como salida adicional se proporciona un tablero para el atacante que permite hacer un seguimiento de los ataques realizados

(d)
Se implementaron las funciones propuestas en la primera parte del trabajo y se incorporaron otras para hacer más modular el programa y que así sea más fácil de desarrollar y entender

"""

```
import random
import os

#-----------------------------------DEFINICIÓN DE VARIABLES Y ESTRUCTURAS-------------------------------

carga_protones = 40 #carga inicial de protones del atacante
carga_electrones = 0 #carga incial de electrones de los escudos

#se genera el diccionario para contener la información del ataque
ataque = dict.fromkeys(("fila","columna","protones"))

#se crea el tablero sin valores
tablero = {"A":["-","-","-","-"],"B":["-","-","-","-"],"C":["-","-","-","-"],"D":["-","-","-","-"]}

#tablero para los ataques del atacante
atacante = {"A":[0,0,0,0],"B":[0,0,0,0],"C":[0,0,0,0],"D":[0,0,0,0]}
#---------------------------------------DEFINICIÓN DE FUNCIONES-----------------------------------
#función para generar los escudos y las cargas de los mismos en el tablero de forma aleatoria
def genera_partida():
        #se indica que se trabaja con variables globales
        global tablero
        global carga_electrones

        #se define aleatoriamente si la nave es horizontal o vertical
        orientacion = random.choice(["horizontal","vertical"])

        #se define las posiciones de los escudos y se actualiza el tablero
```

```
        if orientacion == "horizontal":
                fila = random.choice(list(tablero.keys())) #fila del escudo central
                columna = random.randint(2, 3) #columna del escudo central

                #se llenan las posiciones del tablero obtenidas con valores aleatorios del 1 al
9
                e1 = tablero[fila][columna - 1] = random.randint(1, 9)
                e2 = tablero[fila][(columna - 1) - 1] = random.randint(1, 9)
                e3 = tablero[fila][(columna - 1) + 1] = random.randint(1, 9)
        else:
                fila = random.choice(["B","C"]) #fila del escudo central
                columna = random.randint(1, 4) #columna del escudo central

                #se llenan las posiciones del tablero obtenidas con valores aleatorios del 1 al
9
                e1 = tablero[fila][columna - 1] = random.randint(1, 9)
                e2 = tablero[chr(ord(fila) + 1)][columna - 1] = random.randint(1, 9)
                e3 = tablero[chr(ord(fila) - 1)][columna - 1] = random.randint(1, 9)

        #actualiza la carga de electrones
        carga_electrones += (e1 + e2 + e3)
def imprime_tablero(tablero):
        #función para imprimir el contenido del tablero
        print(f"\t\t1\t2\t3\t4")
        for fila in tablero.keys():
                print(f"\t{fila}", end = "")
                for celda in tablero[fila]:
                        print(f"\t{celda}", end = "")
                print()
def ataque_correcto():
        #función que determina si el ataque ingresado es correcto
        return (ataque["fila"].upper() in list(tablero.keys()))and(ataque["columna"] in range(1,
5))and(ataque["protones"] in range(1, 10))
def insertar_ataque():
        #función para pedir las coordenadas del ataque y la carga de protones
        #usamos la variable global de ataque
        global ataque
```

```
        #bandera para indicar si el ataque ingresado es correcto o no
        flag = False
        first_entry = True

        while not flag:

                #entra al siguiente condicional si no es la primera iteración del while
                if not first_entry:
                        input("\n\nAtaque incorrecto, presione una tecla para ingresar un
nuevo ataque: ")
                        #limpia pantalla
                        os.system("clear")

                print(f"\nCARGA DE PROTONES EN EL REACTOR {carga_protones}")
                print("\n\nCOORDENADAS DEL ATAQUE: ")

                ataque["fila"] = input("\tFILA(A, B, C, D): ")
                ataque["columna"] = int(input("\tCOLUMNA (1, 2, 3, 4): "))
                ataque["protones"] = int(input("\tCARGA DE PROTONES (1-9): "))

                #determina si el ataque ingresado es correcto
                flag = ataque_correcto() and (ataque["protones"] <= carga_protones)

                first_entry = False
        #limpia pantalla
        os.system("clear")
def terminar_juego():
        #función para verificar si se está en condiciones de terminar la partida o no
        return (carga_protones == 0) or (carga_electrones == 0)
#----------------------------------------SE GENERA LA PARTIDA-----------------------------------
def acierto():
        return not(tablero[ataque["fila"].upper()][ataque["columna"] - 1] == '-')

def efectivo():
        return ataque["protones"] <= tablero[ataque["fila"].upper()][ataque["columna"] - 1]
def neutralizado():
        flag = (tablero[ataque["fila"].upper()][ataque["columna"] - 1] == 0)
        return "neutralizado" if flag else "no neutralizado"
```

```
def ganador():
        return((carga_electrones == 0)and(carga_protones > 0))

#--------------------------------------SE GENERAL EL TABLERO-----------------------------------------

genera_partida()

#-------------------------------------PROCESAMIENTO DE LA PARTIDA---------------------------

print("-----------------------------------COMIENZO DEL JUEGO-----------------------------------")

#se juega hasta que se den las condiciones de terminar
while(not terminar_juego()):

        #se piden las coordenadas del ataque junto con la carga del disparo
        #las coordenadas y la carga del disparo se piden hasta que sean valores correctos
        insertar_ataque()

        #se muestra el ataque
        print("-----------------------------------ATAQUE ACUMULADO-----------------------------------")
        atacante[ataque["fila"].upper()][ataque["columna"] - 1] += ataque["protones"]
        imprime_tablero(atacante)
        print("\n-------------------------------------------------------------------------------------")

        #resta los protones del ataque a la carga de protones restantes
        carga_protones -= ataque["protones"]

        if acierto():
                if efectivo():
                        tablero[ataque["fila"].upper()][ataque["columna"] - 1] -=
ataque["protones"]
                        carga_electrones -= ataque["protones"]
                        print(f"Ataque efectivo - Escudo {neutralizado()} - Carga restante de
electrones igual a {carga_electrones}")
                else:
                        print(f"Ataque no efectivo - Escudo {neutralizado()} - Carga restante de
electrones igual a {carga_electrones}")
        else:
```

```
                print("\n\t\tESPACIO!")

        print(f"\nCARGA DE PROTONES EN EL REACTOR {carga_protones}")
        input("\n\nPresione una tecla: ")
        os.system("clear")
if ganador():
        print("\tATAQUE EXITOSO, NAVE NEUTRALIZADA!!!!")

print("\n----------------------------------------------NAVE----------------------------------------------")
        imprime_tablero(tablero)
        print("\n-------------------------------------ATAQUE ACUMULADO-------------------------")
        imprime_tablero(atacante)
        print(f"\nCARGA DE PROTONES EN EL REACTOR {carga_protones}")
else:
        print("\tATAQUE FALLIDO, LA NAVE SIGUE EN PIE!!")

print("\n----------------------------------------------NAVE----------------------------------------------")
        imprime_tablero(tablero)
        print("\n-------------------------------------ATAQUE ACUMULADO----------------------")
        imprime_tablero(atacante)
        print(f"\nCARGA DE PROTONES EN EL REACTOR {carga_protones}")
```

EJERCICIO 6

Escriba un programa que abra el siguiente archivo de texto (a descargar de: http://bit.ly/noseculpeanadie) y retorne las siguientes estadísticas:

(a) Cantidad de caracteres.

(b) Cantidad de oraciones. Una oración culmina con punto.

(c) Cantidad de palabras. Una palabra se separa de la otra por un espacio.

(d) La/s palabra/s de más de 6 caracteres que más veces se repite en el texto.

```
def cuenta_repetidas(p, palabras):
  # Cuenta las palabras de la lista
  return {p:palabras.count(p)}


def function(nombre_archivo):    ##Definimos una función para el filtrado de palabras repetidas
  # Abre y lee archivo txt
  with open(nombre_archivo, 'r') as f:
     texto = f.read()


  # Lista de caracteres no deseados
  no_deseados = [',','.','-','¡','!','¿','?','(',')','"',"'","0","1","2","3","4","5","6","7","8","9"]
  for char in no_deseados:
     texto = texto.replace(char,'')


  # Crea lista de palabras convertidas en minúscula y quitamos de la misma a aquellas de longitud menor a 6 caracteres
  palabras = texto.lower().split()
  a = len(palabras)-1
  for q in range(a+1):
    if (len(palabras[a-q])<6):
      palabras.pop(a-q)


  # Obtiene una lista de diccionarios con key: palabra y value: n. de veces repetida
  dict_list = [cuenta_repetidas(p, palabras) for p in palabras]


  # Elimina los resultados repetidos
  resultados = [i for n, i in enumerate(dict_list) if i not in dict_list[n + 1:]]


  # Ordena los resultados
```

```
  res_ordenados = sorted(resultados, key=lambda d: list(d.values()), reverse=True)

  # Obtiene los primeros 10 resultados
  primeros10 = res_ordenados[:10]

  # Imprime los primeros 10 resultados
  print("\nLas 10 palabras más repetidas, cuya longitud de cadena es mayor que 6, fueron:
\n")
  for d in primeros10:
      for k, v in d.items():
          print(k, v, sep=': ')

nom_archivo='No se culpe a nadie.txt'
with open(nom_archivo,'r') as file:
  caracteres=0
  oraciones=0
  palabras=1
  i=1

  for line in file:
    caracteres+=len(line)          #Cuenta los caracteres, los saltos de línea y puntos
    for word in line:
      if '.' in word:
        oraciones+=1
      if ' ' in word:
        palabras+=1


print("La cantidad de caracteres que contiene el archivo es: ", caracteres)                #(item
a)
print("La cantidad de oraciones que contiene el archivo es: ", oraciones)                  #(item
b)
print("La cantidad de palabras que contiene el archivo es: ", palabras)                    #(item
c)
function(nom_archivo)                                                          #(item d)
file.close()
```

EJERCICIO 7

Dada la siguiente estructura de datos que representa un listado hipotético de alumnos de esta cátedra:

```
informatica = [
 {
 'apellido': 'Aronofsky',
 'entregoTPs?': True,
 'notas': [6,7,6],
 },

 {
 'apellido': 'Nolan',
 'entregoTPs?': True,
 'notas': [7,3,6],
 'recuperatorios':[6],
 },
{
 'apellido': 'Bielinski',
 'entregoTPs?': False,
 'notas': [7,8,8],
 },
 {
 'apellido': 'Miyazaki',
 'entregoTPs?': True,
 'notas': [4,4,6],
 'recuperatorios':[4, 7],
 }
]
```

Escriba un programa que:

(a) Agregue un valor a cada alumno con clave ‘condición’ que indique si ‘recursa’, ‘regular’ o ‘promoción’ siguiendo las reglas de evaluación descritas en el programa de la cátedra

(b) Imprima por pantalla un listado que indique el nombre y la condición de cada alumno en forma de lista.

(c) Indique un resumen estadístico: Total de alumnos, cantidad de alumnos por condición y promedio general de todas las notas de todos los alumnos.

```
informatica = [
{
'apellido': 'Aronofsky',
'entregoTPs?': True,
'notas': [6,7,6],
'recuperatorios':[0],
},

{
'apellido': 'Nolan',
'entregoTPs?': True,
'notas': [7,3,6],
'recuperatorios':[6],
},
{
'apellido': 'Bielinski',
'entregoTPs?': False,
'notas': [7,8,8],
'recuperatorios':[0],
},
{
'apellido': 'Miyazaki',
'entregoTPs?': True,
'notas': [4,4,6],
'recuperatorios':[4, 7],
},
{
'apellido': 'Arito',
'entregoTPs?': True,
'notas': [7,3,8],
```

```
'recuperatorios':[8],
}
]

print("Apellido          Condicion")
estadisticosAlumnos={
'cantidad':0,
'recursantes':0,
'regulares':0,
'promocionan':0,
'cant_notas':0,
'suma_notas':0,
}
for alumno in informatica:
  aprobadas=0
  sumapromo=0
  estadisticosAlumnos['cantidad']+=1
  for nota in alumno['notas']:
    estadisticosAlumnos['cant_notas']+=1
    estadisticosAlumnos['suma_notas']+=nota
    if nota>=6:
      aprobadas+=1
      sumapromo+=nota*0.3
  if(alumno['notas'][2]>=6):
    sumapromo+=alumno['notas'][2]*0.1
  for nota in alumno['recuperatorios']:
    estadisticosAlumnos['cant_notas']+=1
    estadisticosAlumnos['suma_notas']+=nota
    if nota>=6:
      aprobadas+=1
      sumapromo+=nota*0.3
  if(not alumno['entregoTPs?'] or aprobadas<3):
    alumno['condicion']='recursa'
    estadisticosAlumnos['recursantes']+=1
  elif(sumapromo<7):
    alumno['condicion']='regular'
    estadisticosAlumnos['regulares']+=1
```

```
  else:
    alumno['condicion']='promociona'
    estadisticosAlumnos['promocionan']+=1
  print(alumno['apellido']," ",alumno['condicion'])
print("\n")
print("Total Alumnos: ",estadisticosAlumnos['cantidad'])
print("Total Recursan: ",estadisticosAlumnos['recursantes'])
print("Total Regulares: ",estadisticosAlumnos['regulares'])
print("Total Promocionan: ",estadisticosAlumnos['promocionan'])
print("Promedio General:
","{:.2f}".format(estadisticosAlumnos['suma_notas']/estadisticosAlumnos['cant_notas']))
```

EJERCICIO 8

"MiTorneoAmateur" es una página web que asiste a la gestión de torneos deportivos. En dicha página el usuario crea una liga, asignándole un nombre, y luego asocia los equipos. Una vez finalizada la creación de la liga, se carga el resultado de todos los encuentros entre los equipos y se va actualizando el puntaje de cada equipo: los partidos jugados, la cantidad de partidos ganados, la cantidad de empatados, los partidos perdidos y los puntos. Por ejemplo, suponiendo una liga de 4 equipos, y luego de la carga de los resultados, se indica una tabla de posiciones así:

Nombre de Liga: *UnCuyo Liga Mecatrónica*

	Jugados	Ganados	Empatados	Perdidos	Puntos
Monty Pyhton's	3	3	0	0	9
Robóticos	3	1	1	0	4
Ubuntines	3	1	0	0	3
Mecatronics	3	0	1	2	1

Cantidad de partidos jugados: *9*

Sobre las reglas de la liga y funcionalidades del sistema:

- Todos los equipos juegan contra todos los adversarios una sola vez.
- Los partidos pueden ser suspendidos o tienen un resultado. Si son suspendidos no registran estadística alguna.
- El puntaje por encuentro es el siguiente: 3 puntos por ganar el encuentro, 1 punto por empate y 0 por derrota.
- El sistema no recuerda los resultados individuales de cada partido, solo mantiene la tabla de posiciones. La tabla principal se muestra una vez al culminar la carga de los encuentros.
- El resultado de la tabla se presenta ordenado por puntos. En el caso que dos equipos posean el mismo puntaje, se considera en segunda instancia que equipo tiene la mejor diferencia de goles (esto es la resta entre goles anotados y goles recibidos).
- Luego de imprimir la tabla se debe indicar además la cantidad de partidos.

```
decision = "si"
equipos = []
ganados = []
empatados = []
perdidos = []
tabla = []
pt_equal = 0
bandera = 1
goles = []
p = 0

def puntos(ganados, empatados):
  pt = ganados * 3 + empatados
  return pt

def resta(anotados, recibidos):
  if anotados > recibidos:
    dif = anotados - recibidos
  else:
    dif = recibidos - anotados
  return dif

def total(num):
    tot = 0
    for i in range(1, num):
        tot = tot + (num - i)
    return tot

titulo = input("Ingrese el nombre de la Liga: ")

while decision == "si" or decision == "SI" or decision == "sI" or decision == "Si":
  equipo = input("\nRegistre un equipo: ")
  equipos.append(equipo)
  decision = input("Ingrese <si> si desea registrar otro equipo: ")
print("\nEquipos:")
print(f"{equipos}")
ppor_equipo = len(equipos)-1;
```

```
print("\nCada equipo tendrá asignado ",ppor_equipo, " partidos")

partidost = total(len(equipos))

for i in range(0, len(equipos)):
  tabla.append([0] * 7)

for i in range(0, len(equipos)):
  g = input("\nPartidos ganados de " + equipos[i] + ": ")
  ganados.append(g)
  e = input("Partidos empatados de " + equipos[i] + ": ")
  empatados.append(e)
  p = input("Partidos perdidos de " + equipos[i] + ": ")
  perdidos.append(p)
  if (int(g) + int(e) + int(p)) > len(equipos) - 1:
    bandera = 0
    break
if bandera == 1:
  suspendidos = input("\nCantidad de partidos suspendidos al final de toda Liga: ")
  if int(suspendidos) > partidost:
    bandera = 3

if bandera == 1:
  for i in range(0, len(equipos)):
    for j in range(0, 7):
      if j == 0:
        tabla[i][j] = equipos[i]
      elif j == 1:
        tabla[i][j] = len(equipos) - 1
      elif j == 2:
        tabla[i][j] = ganados[i]
      elif j == 3:
        tabla[i][j] = empatados[i]
      elif j == 4:
        tabla[i][j] = perdidos[i]
      elif j == 5:
        puntaje = puntos(int(ganados[i]), int(empatados[i]))
```

```
        tabla[i][j] = puntaje

  for i in range(0, len(equipos)):
    value = tabla[i][5]
    for j in range(0, len(equipos)):
      if tabla[j][0] != tabla[i][0]:
        comp = tabla[j][5]
        if value == comp:
          pt_equal = 1

  if pt_equal == 1:
    for i in range(0, len(equipos) + 1):
      goles.append([0] * 6)

    for i in range(0, len(equipos)):
      goles[0][i + 1] = equipos[i]
      goles[i + 1][0] = equipos[i]

    print("\nSepare los siguintes resultados de los partidos con -\n")
    for j in range(1, len(equipos) + 1):
        for i in range(1, len(equipos) + 1):
          if i > j:
            p = int(p) + 1
            goles[i][j], goles[j][i] = input("Partido " + str(p) + " " + str(goles[0][j]) + " vs " +
str(goles[i][0]) + ": ").split("-")
          if i == j:
            goles[i][j] = 0

    for j in range(1, len(equipos) + 1):
      a = 0
      r = 0
      for i in range(1, len(equipos) + 1):
        a = a + int(goles[j][i])
        r = r + int(goles[i][j])
        dif = resta(a, r)
      tabla[j - 1][6] = dif

  print("\nLiga ", titulo)
```

```
  if pt_equal == 1:
    tabla.sort(reverse=True, key=lambda x: x[6])
    tabla.sort(reverse=True, key=lambda x: x[5])
    tabla.insert(0, ["T", "P", "V", "E", "D", "C", "G"])
    print("\nT: Equipos\nP: Partidas jugadas\nV: Victorias\nE: Empates\nD: Derrotas\nC: Clasificación\nG: Diferencia de goles anotados y recibidos")
    for i in range(0, len(equipos) + 1):
      print("\n")
      for j in range(0, 7):
        print(tabla[i][j],end="   ")
  else:
    tabla.sort(reverse=True, key=lambda x: x[5])
    tabla.insert(0, ["T", "P", "V", "E", "D", "C"])
    print("\nT: Equipos\nP: Partidas jugadas\nV: Victorias\nE: Empates\nD: Derrotas\nC: Clasificación")
    for i in range(0, len(equipos) + 1):
      print("\n")
      for j in range(0, 6):
        print(tabla[i][j],end="   ")
  print("\n\nCantidad de partidos jugados: ", partidost - int(suspendidos))
elif bandera == 0:
  print("\nLa cantidad de partidos ganados, empatados y perdidos no puede superar la cantidad de partidos jugados por el equipo")
elif bandera == 3:
  print("\nLa cantidad de partidos suspendidos no pueden ser mayores al total")
```

EJERCICIO 9

La empresa de mantenimiento “La pinturita” desea calcular la cantidad de latas de pintura necesarias del mismo color para pintar las paredes de todas las habitaciones de una casa. Se conoce la siguiente información: Las puertas son de 0.75 x 2.00 mts (ancho x alto) y las ventanas son de 1.20 x 1.50 mts. La pintura tiene las siguientes características:

Tipo de Latex	Cada litro rinde (por mano)	Viene en latas de (litros)
Mate	14 m^2	1, 4, 10 y 20

La información variable consiste de:

- El ancho, largo y alto de cada habitación.
- Cantidad de cada tipo de aberturas en cada habitación.
- Cantidad de manos a pintar (una “mano” representa cubrir completamente la superficie con pintura).

El programa debe determinar la cantidad de latas de pintura a utilizar de manera tal que se minimice el costo total, informando al usuario dicho costo. El costo de cada lata es el siguiente: la lata de 1 litro cuesta $50, la de 4 cuesta $170, la de 10 cuesta $400 y la de 20 litros cuesta $700. Observe que cuantas menos latas se compren menos se paga, ya que, por ejemplo, una lata de 4 lts cuesta menos que 4 latas de 1 lts.

```
import math

puerta_alto=2
puerta_ancho=0.75
ventana_alto=1.5
ventana_ancho=1.2
cant20=cant10=cant4=cant1=precio20=precio10=precio4=precio1=0
def latas(litros):
  global cant20, cant10,cant4,cant1,precio20,precio10,precio4,precio1
  resta20=litros-20
  if(resta20>=0):
    latas(resta20)
    cant20= cant20+1
    precio20=precio20+700
  else:
    resta10=litros-10
    if (resta10>=0):
      latas(resta10)
      cant10=cant10+1
      precio10=precio10+400
    else:
      resta4=litros-4
      if(resta4>=0):
        latas(resta4)
        cant4=cant4+1
        precio4=precio4+170
      else:
        resta1=litros-1
        if(resta1>=0):
          latas(resta1)
          cant1=cant1+1
          precio1=precio1+50
```

```
def litros(ancho, alto, largo, cant_vent, cant_puertas, manos):  ##suponiendo que las habitaciones son rectangulares cuanto mucho
  superficie_paredes= 2*alto*ancho + 2*alto*largo
                    superficie_aberturas=  cant_vent*ventana_alto*ventana_ancho  +  cant_puertas*puerta_alto*puerta_ancho
  deltasuperficie=(superficie_paredes-superficie_aberturas)
  if(deltasuperficie<=0):


    return -1
  else:
    superficie_a_pintar= deltasuperficie*manos
    l = math.ceil(superficie_a_pintar/14)
    print("\nlitros necesarios de pintura: ", l)
    return l




l=-1
while l==-1:
  puertas = int(input("Ingrese cantidad de puertas: "))
  ventanas = int(input("Ingrese cantidad de ventanas: "))
  ancho = float(input("Ingrese ancho (m) de la habitación: "))
  largo = float(input("Ingrese largo (m) de la habitación: "))
  alto = float(input("Ingrese alto (m) de la habitación: "))
  manos = int(input("Ingrese cantidad de manos a pintar: "))
  l=litros(ancho,alto,largo,ventanas,puertas,manos)
  if l==-1:
    print("error: hay mas superficie ocupada por aberturas que por pared")

latas(l)
totallatas=cant20+cant10+cant4+cant1
preciototal= precio20+precio10+precio4+precio1
print("\n Latas a utilizar = ", totallatas)
print("\n Latas de 20L = ", cant20, " con un precio de $", precio20)
print("\n Latas de 10L = ", cant10, " con un precio de $", precio10)
print("\n Latas de 4L = ", cant4, " con un precio de $", precio4)
print("\n Latas de 1L = ", cant1, " con un precio de $", precio1)
print("Total = $",preciototal)
```