UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA - UTEC

Tarea 1.2

Fecha de inicio: *12 de Setiembre, 2022*
Fecha de entrega: *25 de Setiembre, 2022*

Curso: *Programación 1 (1100)* – Entrega mediante: *Gradescope*

Indicaciones generales

1. Recuerda que la tarea es **individual**. Los casos de copia/plagio serán sancionados con nota cero (0) en la asignatura.
2. (a) La fecha límite de entrega es el **domingo 25 de setiembre a las 23:59 hrs**.
   (b) Es altamente recomendable no esperar hasta la última hora.
   (c) *Gradescope* desactivará automáticamente los envíos pasada dicha hora límite.
   (d) **No se aceptarán entregas atrasadas ni entregadas por otros medios.**
3. Revisa bien lo que entregas, aunque en esta oportunidad podrás entregar ilimitadas veces la tarea, la última enviada será la evaluada.
4. Recuerda que *Gradescope* corrige automáticamente tu entrega. Dicha plataforma mostrará si has realizado correctamente las pruebas y mostrará algunos mensajes en color verde. Puedes ver un ejemplo de este caso en el anexo 1.
5. Es posible que hayas subido tu entrega pero hayas modificado algo que no se debió en el template. En ese escenario, Gradescope te mostrará algunos mensajes de error. Puedes ver un ejemplo de esto en el anexo 2.

Gradescope

1. Nosotros les proporcionaremos un código base de donde deberán partir para completar dicho ejercicio. Este archivo es llamado `solution.py` y lo encontrarán en la indicación de la tarea en CANVAS.
2. Al finalizar, **solo** subir el archivo `solution.py` (NO cambiar el nombre del archivo y NO comprimirlo).

3. Cada pregunta tiene diversos casos de prueba. Para obtener la nota completa en una pregunta, el algoritmo debe obtener la respuesta correcta en dichos casos de prueba.

4. Si un caso de prueba falla, visualizarán un mensaje de error con sugerencias. **Lee el error**, revisa el código e inténtalo de nuevo.

5. Los input de los casos de prueba son confidenciales.

Indicaciones específicas

1. En el anexo 1, se puede ver la plantilla de código.

2. Ustedes deben escribir dentro de la sección y a la misma altura de donde esta escrito *"Código comienza aquí"*. Además, no deben modificar nada debajo de *"Código acaba aquí"*. Recuerden tener cuidado con las indentaciones.

3. Los input del ejercicio se encuentran en la plantilla. Recuerden usar estas variables para resolver el ejercicio.

4. La respuesta del ejercicio debe ser impresa **especificamente** con *print()* para que Gradescope la tome en consideración. En el caso de funciones, esta debe ser retornada, según especifique la plantilla otorgada.

5. Al momento de la impresión de la respuesta, no adicionar texto. Imprimir **únicamente** el resultado que pide el ejercicio. De la misma manera con las funciones.

Caso: Protege la Linea Sagrada del Tiempo - (15 pts)

La Autoridad de Variacion Temporal(AVT) necesita tu ayuda para mantener el orden de la Sagrada Linea del Tiempo. Para ello, debes crear un programa que basado en los viajes que Sylvie y Loki realizan, determines el estado de la linea temporal. En cada viaje temporal Loki y Sylvie son interceptados por soldados de la AVT, producto de estos encuentros ellos pueden ganar o perder puntos los cuales les permiten seguir viajando.

- El numero de viajes que Loki y Sylvie realizan juntos esta representado por un numero ***N***.

- Durante cada encuentro Loki y Sylvie pueden ganar y/o perder puntos.

- Cada viaje en el tiempo tiene un conjunto de puntos ganados y perdidos.

- No necesariamente todos los conjuntos de puntajes de cada viaje tienen la misma cantidad de elementos.

Parámetros del programa

1. Un número entero ***N*** que representan la cantidad de viajes realizados por Loki y Sylvie.
2. Una lista ***viajes*** de longitud ***N*** que representa los años a los que Loki y Sylvie han viajado.
3. Una lista ***P*** de longitud ***N*** donde cada elemento p_i la longitud del conjunto de puntos ganados o perdidos en el i-esimo viaje.
4. Una lista de listas ***puntajes***. Donde la i-esima lista representa el conjunto de puntajes obtenidos en el i-esimo viaje y tiene tamaño/longitud p_i.
5. **Recuerde que usted debe parsear y castear el input a los tipos de dato que considere necesario.**

Formato de entrada: :

```
N
viajes = [v_1, v_2, ... v_i, ..., v_N]
P = [p_1, p_2, ..., p_N]
puntajes = [[]_1, []_2, ..., []_N]
```

Ejemplo de entrada: :

```
N=5
viajes=[1980, 2005, 1821, 2020, 1457]
P=[3, 4, 1, 2, 2]
puntajes=[[40, -20, 5], [5, 15, 7, -13], [10], [8, -4], [17, -8]]
```

Entrega de resultados

1. Si el número de viajes ***N*** es igual a 0. Usted debe imprimir "La línea del tiempo está libre de los viajes de Loki y Sylvie."
2. Si el número de viajes ***N*** es menor a 0. Usted debe imprimir "ERROR, no existe información acerca de los viajes de Loki y Sylvie."
3. Si cualquier cantidad de puntos por encuentro (P_i) es un número menor o igual a 0. Usted debe imprimir "ERROR, no pueden existir viajes sin puntos.". Y no ejecutará el diagnóstico del estado de la línea del tiempo.
4. En caso de insertar 2 años iguales, se debe imprimir "PELIGRO, se ha viajado al mismo año 2 veces, envíe mas soldados en su captura."

5. En el caso de que la suma de puntos entre todos los viajes sea positiva o 0, usted debe imprimir "Loki y Sylvie pueden seguir viajando, envíe mas soldados en su captura.".

6. De lo contrario, debe imprimir "Los soldados de la AVT han logrado atrapar a Loki y Sylvie luego de ***N*** viajes en el año ***X***.". Donde ***X*** es el último año a donde Loki y Sylvie pudieron escapar y ***N*** es el número de viajes indicados al inicio.

Usted solo debe imprimir un único mensaje, en el caso de que usted detecte múltiples casos distintos al mismo tiempo, su programa debe imprimir el mensaje con menor índice según la numeración presentada anteriormente (números más pequeños simboliza mayor prioridad). Por ejemplo, si usted detecta el caso de error de la regla 2, pero a la vez el caso de error de la regla 3. Solo debe imprimir el mensaje "ERROR, no existe información acerca de los viajes de Loki y Sylvie.". **Tenga en cuenta que las únicas reglas consideradas como error son 2 y 3**.

Ejemplo 1.

Parámetros :

```
N=0
codigos=[]
M=[]
notas=[]
```

Resultado impreso :

```
  La línea del tiempo está libre de los viajes de Loki y Sylvie.
```

Explicación El valor de N es igual a 0, por ello, la línea del tiempo esta libre de peligro.

Ejemplo 2.

Parámetros :

```
N=-1
viajes=[]
P=[]
puntajes=[]
```

Resultado impreso :

```
ERROR, no existe información acerca de los viajes de Loki y Sylvie.
```

Explicación El valor de N es menor a 0, por ello, se detectó un error (información faltante).

Ejemplo 3.

Parámetros :

```
N=4
viajes=[1717,1669,1781,1707]
P=[2,-1,2,0]
puntajes=[[20,19],[],[10,11],[]]
```

Resultado impreso :

```
ERROR no pueden existir viajes sin puntos.
```

Explicación El valor de algún P_i es menor o igual a 0, por ello, se detectó un error (viaje sin puntos).

Ejemplo 4.

Parámetros :

```
  N = 5
  viajes=[1463,1780,1999,2000,2050]
  P=[2,3,1,4,1]
  puntajes=[[16,-9],[1,-20,15],[20],[5,-10,20,-30],[13]]
```

Resultado impreso :

```
  Loki y Sylvie pueden seguir viajando, envie mas soldados en su captura
    .
```

Explicación Luego de recibir la entrada. Se debe sumar la cantidad de puntos obtenidos en cada viaje, el resultado de este es 21. Por lo que al haber obtenido un puntaje final positivo, Loki y Sylvie pueden seguir viajando a traves de la Linea del Tiempo.

Ejemplo 5.

Parámetros :

```
  N = 5
  viajes=[1463,1780,1999,2000,2050]
  P=[2,3,1,4,2]
  puntajes=[[16,-9],[1,-20,15],[20],[5,-10,20,-30],[13,-27]]
```

Resultado impreso :

```
Los soldados de la AVT han logrado atrapar a Loki y Sylvie luego de 5
  viajes en el año 2050.
```

Explicación El resultado de la suma de puntos es -6. Por lo que al haber obtenido un puntaje final negativo, Loki y Sylvie lograron ser atrapados por los soldados de la AVT.

Ejemplo 6.

Parámetros :

```
N=5
viajes=[1702,1588,1690,1640,1702]
P=[4,1,3,2,5]
puntajes=[[-20,18,28,20],[67],[-47,3,-4],[22,44],[2,-26,-8,-38,28]]
```

Resultado impreso :

```
PELIGRO se ha viajado al mismo año 2 veces envie mas soldados en su
   captura.
```

Explicación Se insertó 2 veces el año 1702. Por ello se imprime el mensaje de peligro.

1. Anexos

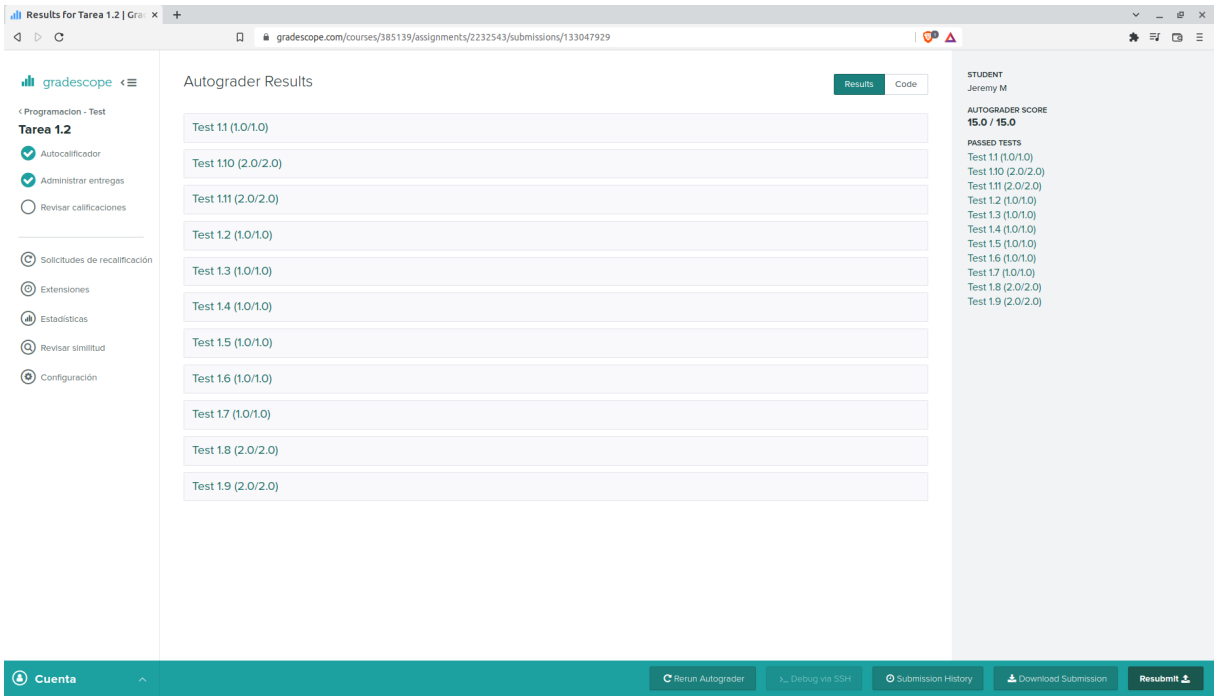

Figure 1: Casos de prueba correctos en Gradescope.

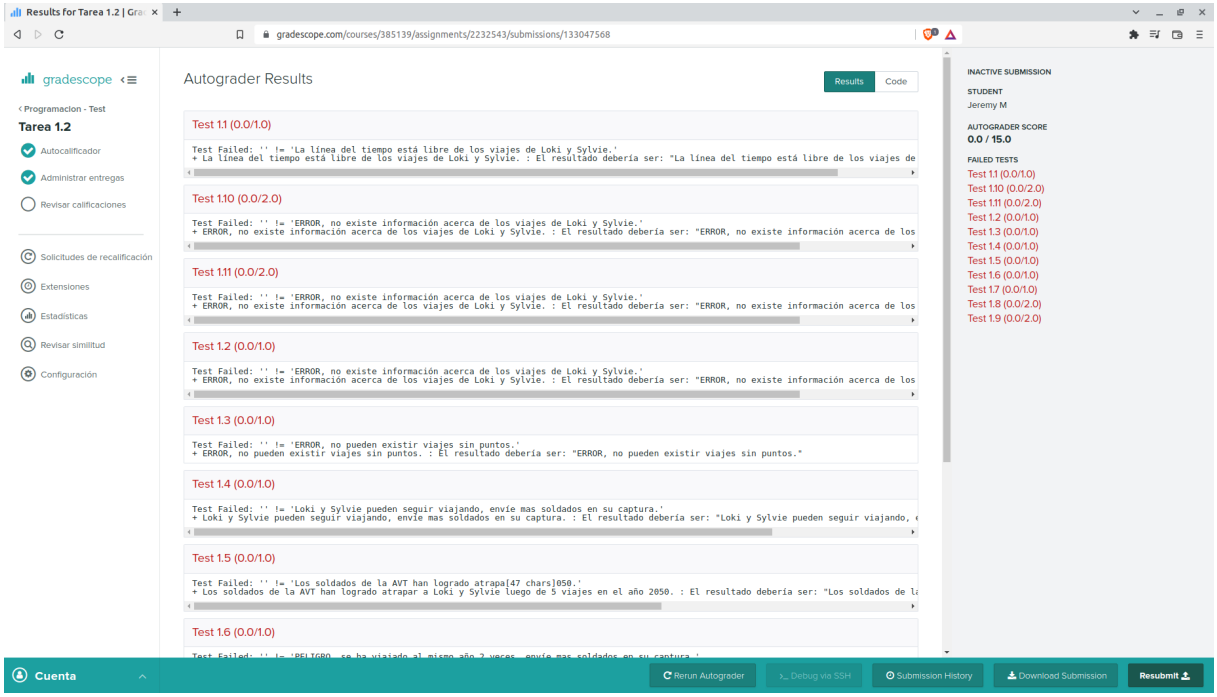

Figure 2: Entrega incorrecta en Gradescope.

```python
# NO MODIFICAR NADA BAJO ESTA LINEA
def main():
# NO MODIFICAR NADA SOBRE ESTA LINEA
    #=======PROTEGE LA SAGRADA LINEA DEL TIEMPO=======
    N = int(input("N = "))
    viajes = input("viajes = ")
    P = input("P = ")
    puntajes = input("puntajes = ")

    # tu solucion empieza aqui
    print("")
    # tu solucion termina aca

# NO MODIFICAR NADA BAJO ESTA LINEA
if __name__ == "__main__":
    main()
# NO MODIFICAR NADA SOBRE ESTA LINEA
```

Listing 1: Template `solution.py`.