RETO 4 – Fundamentos de Programación

Usted es un desarrollador en una compañía que se dedica a desarrollar software para casinos y apuestas, su jefe le encargó programar un juego de cartas para dos jugadores, en este se usa una baraja estándar de naipes de 52 cartas, el jugador 1 comienza sacando una carta, luego el jugador 2 continúa y así sucesivamente hasta que no quede ninguna carta en la baraja.

La baraja estándar tiene 13 tipos de cartas: ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K'] donde cada tipo aparece 4 veces, una por palo, por tanto hay 4 palos (trébol, pica, diamante, corazón) pero solo nos interesa la cantidad de veces que aparece cada tipo de carta, en este caso 4. Las cartas ['A', 'J', 'Q', 'K'] se conocen como cartas altas. Cuando un jugador toma una carta alta, tiene la posibilidad de ganar puntos de la siguiente manera:

- Si el jugador toma una carta **A** y la siguiente carta en la baraja no es una carta alta, gana 1 punto.
- Si el jugador toma una carta **J** y ninguna de las dos cartas que siguen en la baraja son una carta alta, gana 2 puntos.
- Si el jugador toma una carta **Q** y ninguna de las tres cartas que siguen en la baraja son una carta alta, gana 3 puntos.
- Si el jugador toma una carta **K** y ninguna de las cuatro cartas que siguen en la baraja son una carta alta, gana 4 puntos.

Si hay menos cartas de las necesarias para ganar puntos, estos no se obtendrán.

Su función solución $jugar(baraja)$ tendrá como entrada una baraja de cartas, como la descrita anteriormente, desordenada, además, debe de retornar los puntajes del jugador 1 y 2 respectivamente.

Por otra parte, en su función solución debe de invocar una función booleana llamada $tiene\_cartas\_altas(cartas\_siguientes)$, como puede ver, para calcular los puntajes hay diferentes condiciones que tienen el mismo comportamiento de verificar si en las cartas siguientes, que tienen una cantidad variable, hay alguna carta alta, por lo tanto este comportamiento se puede agrupar en una función lo que evita la repetición de código y promueve las buenas prácticas.

Ejemplo

Observe que a la función *juego* le entra una baraja de longitud 52 de tipo tupla y le salen los puntajes de los jugadores 1 y 2 respectivamente:

```
baraja = ('Q', '3', 'Q', '2', '2', '3',...
juego(baraja)
```

```
(21, 9)
```

```
cartas_siguientes = baraja[1:4]
cartas_siguientes
```

```
('3', 'Q', '2')
```

```
tiene_cartas_altas(cartas_siguientes)
```

```
True
```

Note que se está imprimiendo el retorno de la función, en la definición de esta **NO** se imprime nada.

Además, que el jugador 1 saca la carta **Q** que es una carta alta, sabemos que si ninguna de las siguientes tres cartas es alta entonces el jugador sumará 3 puntos, observe que en las cartas siguientes se encuentra otra carta alta **Q**, por tanto el jugador no recibirá puntaje. Observe el uso de la función *tiene_cartas_altas*, se le ingresan las cartas siguientes que se quieren revisar usando **SLICING**, la función retorna el valor booleano **True** porque las cartas siguientes contienen a **Q** como ya vimos.

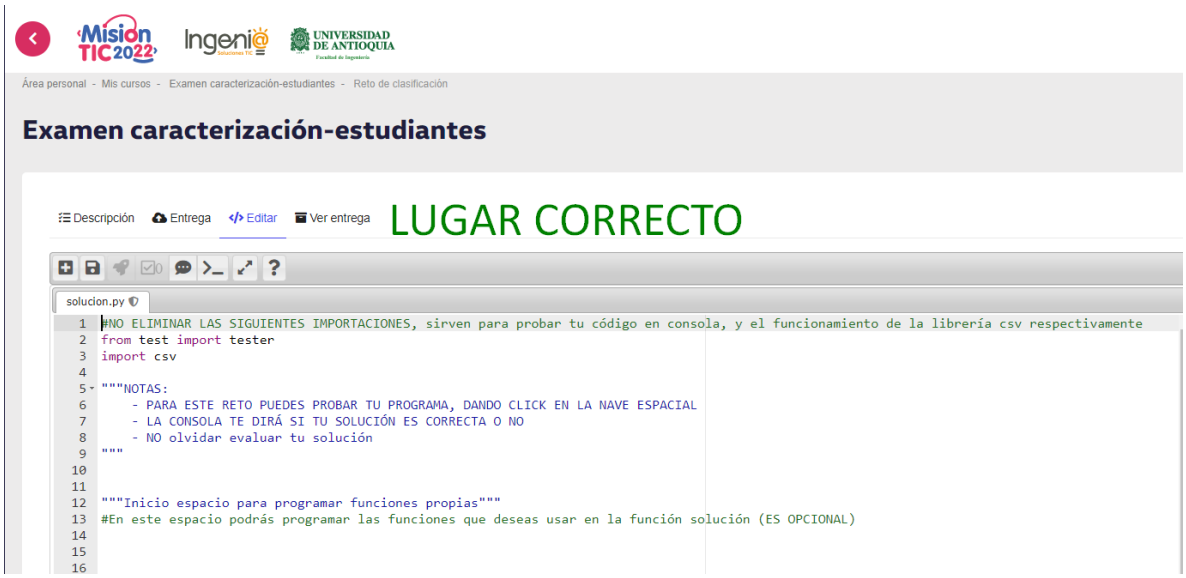
Tenga en cuenta que si la baraja termina en: ...K, 1, 2, 3. El jugador que sacó la carta **K** no debe recibir puntaje, lo recibiría si a **K** le siguieran 4 cartas no altas, pero en este caso solo le siguen 3, de forma similar funciona con las demás cartas altas.

NOTA ACLARATORIA

Se recomienda desarrollar la prueba en un IDE como G Colab, VSCode, PyCharm, Spyder, etc. Para esto se puede copiar y pegar el esquema de solución proporcionado en el VPL a su IDE preferido, recuerde que al final debe copiar y pegar el código del IDE a la herramienta VPL, pero **NO** deberá subir archivos, es decir:

Modo incorrecto:

Modo correcto:



TRIPULANTE, ¡MUCHOS ÉXITOS EN EL DESARROLLO DEL RETO 4!