UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA

10110 – Fundamentos de Computación y Programación
10145 – Fundamentos de Programación para Ingeniería

[Ex] EXAMEN

INSTRUCCIONES

Antes de comenzar, **lea atentamente las instrucciones, los requerimientos de entrega y las preguntas de la evaluación**.

- **La evaluación consta de dos preguntas, de igual ponderación.**
- La evaluación es de carácter individual y está sujeta a las normas del código de honor de la asignatura. **Cualquier indicio de intervención de otra persona o de cualquier otro acto sancionable como falta al código de honor significará la calificación mínima al promedio de Teoría.**
- Soluciones que estén *hardcodeadas* para pasar exclusivamente los casos de prueba serán calificadas con puntaje mínimo.
- Tenga en cuenta que si su solución no cumple con la especificación de la pregunta, su evaluación será calificada con nota mínima. Por ejemplo, si la pregunta especifica que la solución debe ser recursiva, aunque la solución pase todos los testcases, si esta no hace uso de recursión, no se le asignará puntaje.
- La solución debe desarrollarse directa y completamente en Python y debe entregarse por UVirtual. Durante la evaluación el estudiante solo tiene permitido el uso de Python (directamente en el IDLE, consultar el enunciado de la evaluación y el resumen de contenidos). Soluciones desarrolladas con otros medios serán calificadas con nota mínima.
- Estudiantes que entreguen fuera de plazo serán calificados con nota mínima en esta evaluación.
- Entregas que se realicen por vías distintas a Campus Virtual serán calificadas con nota mínima. En caso de que la plataforma presente un problema, puede enviar su archivo como respaldo al correo de contacto del profesor de Teoría, y posteriormente subir el archivo a Campus Virtual.
- Estudiantes que entreguen sin seguir los criterios de identificación estipulados en el apartado de "entrega" serán calificados con nota mínima en esta evaluación.
- Considere que cualquier supuesto que el estudiante haga debe ser explicitado en los comentarios de código.
- La subida del archivo es responsabilidad de su autor, por lo que archivos que no estén en el formato estipulado, que vengan corruptos o con problemas para ser leídos no serán revisados.
- **La evaluación (E) se evaluará considerando 50% puntaje de casos de prueba y 50% rúbrica de desarrollo.**

ENTREGA

Para cada pregunta cree un archivo .py con su RUN. Agregue al encabezado de su programa los siguientes datos de identificación. Considere que, de no agregarlos, el puntaje de su pregunta no contará.

```
# FUNDAMENTOS DE PROGRAMACIÓN PARA INGENIERÍA
#FUNDAMENTOS DE COMPUTACIÓN Y PROGRAMACIÓN
# SECCIÓN DEL CURSO:
# PROFESOR DE TEORÍA:
# PROFESOR DE LABORATORIO:
#
# AUTOR
# NOMBRE:
# RUN:
# CARRERA:
```

Suba su solución al apartado especificado en UVirtual en la pestaña “EVALUACIONES”, apartado “[Ex] – PREGUNTA 1” o “[Ex] – PREGUNTA 2” según corresponda.

1.DE NÚMEROS Y MORSE

Uno operadores de inteligencia estratégica militar se están comunicando utilizando código Morse. Esto lo usan para transmitir distintos valores relacionados a las tropas enemigas, a veces distancias, otras veces cantidad de tropas y otros elementos.

Para poder comunicarse, y entender que los mensajes son aliados han determinado un formato, en este cada mensaje es un único string de caracteres raya (`-`) y punto (`.`). Para que el mensaje sea válido, este debe respetar la estructura `<encabezado><mensaje><indicador de final>`. El indicador de inicio de mensaje estará simbolizado por la secuencia raya-punto-raya (`'-.-'`). Luego, el mensaje será una secuencia de caracteres donde 5 símbolos representarán un dígito de un número entero. Finalmente, se tiene un indicador de final de mensaje simbolizado como raya-raya-raya-raya-punto-raya (`'----.-'`). Considera que el mensaje solo traerá números naturales positivos y que estos están codificados como se indica en la Tabla 1.

Tabla 1 – Codificación de números en morse

Número	Codificación
0	`-----`
1	`.----`
2	`..---`
3	`...--`
4	`....-`
5	`.....`
6	`-....`
7	`--...`
8	`---..`
9	`----.`

Deberás construir un programa que deberá validar que el mensaje sea correcto, y en caso de que lo sea deberás entregar el mensaje decodificado. En caso de que el mensaje no cumpla con el formato deberás informarlo e indicar los errores que el mensaje posee.

ENTRADA

La entrada del programa será un único string, representando el mensaje en código Morse recibido. Para solicitarlo deberás hacerlo con el texto `'Ingrese mensaje: '`, ten en cuenta que después del carácter dos puntos existe un espacio. Por ejemplo:

```
Ingrese mensaje: -.------.---------.--------.-
```

SALIDA

Deberás imprimir una salida distinta dependiendo del caso. Si el mensaje recibido no tiene errores deberás informarlo con el texto **`'El numero enviado es: <numero>'`**, donde `<numero>`, es el número

decodificado. Ten en cuenta que el número puede tener más de un dígito y que en caso de comenzar con 0, este debería imprimirse.

En caso de que el mensaje no sea el correcto deberías imprimir:

- **`'El mensaje no comienza correctamente'`**, si el carácter de inicio de mensaje no es el correcto.
- **`'El mensaje no finaliza correctamente'`**, si el carácter de fin del mensaje no es el correcto.
- **`'Largo del mensaje incorrecto'`**, si la cantidad de caracteres del mensaje no cuadra con la cantidad que debería tener. Ten en cuenta que cada número será de 5 caracteres de largo, para esta validación, no se consideran ni los marcados de inicio, ni de final del mensaje.
- **`'Se encontraron caracteres no reconocidos'`**, si dentro del mensaje vienen caracteres que no concuerdan con el alfabeto.

Ten en cuenta que si el mensaje no es válido por más de una de estas condiciones, deberás indicar cada mensaje en una línea distinta, en el mismo orden en el que aquí se presentan. Si el mensaje no tiene el formato adecuado, la revisión de caracteres no reconocidos no debe realizarse.

Por ejemplo, para el caso del ejemplo de entrada, el mensaje no tiene errores, por lo que la salida correcta sería **`'El numero enviado es: 0101'`.**

CONSIDERACIONES

Considera que los strings siempre traerán solo los caracteres guion medio `(-)`, representando la raya y punto `(.)` representando el punto.

Nota que el error de caracteres no reconocidos solo se debe verificar si es que el mensaje tiene una estructura válida, es decir, comienza y termina correctamente y tiene un largo válido.

Para este ejercicio se prohíbe el uso del tipo de dato diccionario y manejo de excepciones usando bloques try/except.

EJEMPLOS

ENTRADA 1

```
Ingrese mensaje: -.-.----..---...------.-
```

SALIDA 1

```
El numero enviado es: 123
```

ENTRADA 2

```
Ingrese mensaje: -.-.----..---.-.------.-
```

SALIDA 2

```
Se encontraron caracteres no reconocidos
```

ENTRADA 3

```
Ingrese mensaje: ---.----..---..---------
```

SALIDA 3

```
El mensaje no comienza correctamente
El mensaje no finaliza correctamente
```

2. ATRAPALOS YA

Pokémon es una franquicia multimedia japonesa creada por Satoshi Tajiri y Ken Sugimori, que comenzó como un videojuego para consolas portátiles de Nintendo. La palabra "Pokémon" es la contracción de "*Pocket Monsters*" (Monstruos de Bolsillo), refiriéndose a las criaturas ficticias que los jugadores capturan y entrenan para que luchen entre sí por diversión, deporte o competencia. La franquicia Pokémon ha crecido enormemente desde su debut en 1996, incluyendo videojuegos, series de televisión, películas, juegos de cartas coleccionables, juguetes y otros productos de *merchandising*. En los videojuegos, los jugadores asumen el papel de entrenadores Pokémon que viajan a través de regiones, capturando y entrenando Pokémon para competir en torneos y enfrentarse a desafíos.

Una de las modalidades de combate en Pokémon es el 6 vs 6, donde dos entrenadores combaten cada uno con un equipo de 6 Pokémon, los cuales se enfrentan 1 vs 1 hasta que uno de los dos equipos quede totalmente incapacitado para seguir luchando.

En esta oportunidad, se te pide crear un programa en Python que simule una batalla Pokemon, a partir de un listado de pokemones (`pokemon.txt`) y un listado de tipos (`tipos.txt`). El programa debe hacer lo siguiente:

1. Obtener desde los archivos `pokemon.txt` y `tipos.txt` las características relevantes para el combate.
2. Solicitar a dos usuarios el ingreso de los pokemon que utilizarán para sus combates. Cada uno utilizará 6.
3. Ya formados los equipos, el programa deberá simular el combate.
4. Para determinar al ganador del combate con su equipo.

Para la simulación de estos combates, se deben enfrentar las criaturas en el orden que fueron seleccionados por los usuarios. El ganador de cada combate se determina según las siguientes reglas:

- Si un pokemon tiene como tipo alguna de las debilidades de su contrincante, este debería ganar la batalla.
- Si ambos tienen debilidades al contrincante, o ninguno tiene debilidades ganará el pokemon que tenga más vocales en su nombre.
- En caso de que el empate persista aún en este caso, se declarará como ganador al pokemon del usuario 2.

Al ganar un combate, el Pokémon vencedor debe mantenerse para el siguiente, y el contrincante debe enviar al siguiente Pokémon de su lista. Gana el Usuario que quede con Pokémon en pie luego de eliminar todos los pokemones del otro usuario.

ENTRADA

Las entradas serán dos documentos de texto y los equipos formados por el los usuarios. El primero llamado `pokemon.txt` que contendrá a los Pókemon con sus respectivos tipos. El segundo documento llamado `tipos.txt` contiene las debilidades de cada tipo.

El archivo `pokemon.txt`, tendrá varias líneas, donde cada línea representa un pokemon seleccionable, de la forma `<pokemon> - <tipo 1>` o `<pokemon> - <tipo 1> / <tipo 2>` según corresponda. Por ejemplo:

```
Diglett - Tierra

Bulbasaur - Planta / Veneno
```

Considera que la cantidad de pokemones que puede traer el archivo `pokemon.txt` podría variar. Ten en cuenta que el usuario siempre ingresará el nombre del pokemon exactamente como está escrito en el archivo.

Por otro lado, el archivo `tipos.txt`, contendrá varias líneas de texto representando los tipos. Cada línea tendrá el formato `<tipo X>: Debil contra <tipo Y>, <tipo Z>`. Donde cada tipo estará separado por comas. Por ejemplo:

```
Roca: Debil contra Agua, Planta, Lucha, Tierra, Acero
```

Considera que si un tipo tiene una única debilidad, el formato será `<tipo X>: Debil contra <tipo Y>`. Por ejemplo:

```
Electrico: Debil contra Tierra
```

Ten en cuenta que los tipos y los mensajes nunca tendrán tildes.

Adicionalmente, deberás solicitar a los usuarios sus pokemones, para esto deberás utilizar el mostrar el mensaje `' Usuario <N°> - Ingrese su equipo Pokemon'` para que el usuario sepa que es su turno de ingresar pokemones y luego `'1: '`, `'2: '`, `'3: '`, `'4: '`, `'5: '` y `'6: '` para ingresar los pokemones correspondientes. Ten en cuenta que todos los mensajes tienen un espacio luego del carácter dos puntos. Un ejemplo de entrada sería:

```
Usuario 1 - Ingrese su equipo Pokemon
1: Pikachu
2: Venasaur
3: Charizard
4: Blastoise
5: Lapras
6: Snorlax

Usuario 2, Ingrese su equipo Pokemon
1: Pidgeot
2: Alakazam
3: Rhydon
4: Arcanine
5: Exeggutor
6: Blastoise
```

SALIDA

Se deberá mostrar la evolución de la batalla en el formato:

```
Combate <N°> - <Pokemon Usuario 1> vs <Pokemon Usuario 2>
Vencedor: <Pokemon vencedor>
```

Dónde `<N°>` es el número de la batalla, `<Pokemon Usuario 1>` es el pokemon activo del usuario 1, `<Pokemon Usuario 2>`es el pokemon activo del usuario 2 y `<Pokemon vencedor>`es el pokemon que gana ese duelo.

Una vez finalizada la batalla deberá mostrarse el mensaje

```
EL VENCEDOR DE ESTA BATALLA ES: Usuario <N°>
Con su team: <Pokemon 1>, <Pokemon 2>, <Pokemon 3>, <Pokemon 4>, <Pokemon 5>,
<Pokemon 6>
```

Donde `<N°>` indica si el ganador es el usuario 1 o 2 y la segunda línea muestra los 6 pokemon de ese equipo separados por comas.

CONSIDERACIONES

- Considere que los nombres en tipos.txt, pokemon.txt y las entradas del usuario siempre serán consistentes y no tendrán errores.
- Para su solución se prohíbe usar diccionarios y listas por comprensión.
- El uso de módulos que tengan que instalarse (como pandas) está prohibido, así como módulos específicamente diseñados para leer archivos con algún formato especial, como csv.
- También se prohíbe el uso de manejo de excepciones con try/except.
- Añadir el contenido de los archivos al código, en lugar de leerlos, se considerará hardcoding y evaluado como tal.

EJEMPLOS

ENTRADA 1

Usuario 1

```
1: Pikachu
2: Venusaur
3: Charizard
4: Blastoise
5: Lapras
6: Snorlax
```

Usuario 2

```
1: Pidgeot
2: Alakazam
3: Rhydon
```

```
4: Arcanine
5: Exeggutor
6: Blastoise
```

SALIDA 1

```
Combate 1 - Pikachu vs Pidgeot
Vencedor: Pikachu
Combate 2 - Pikachu vs Alakazam
Vencedor: Alakazam
Combate 3 - Venusaur vs Alakazam
Vencedor: Alakazam
Combate 4 - Charizard vs Alakazam
Vencedor: Alakazam
Combate 5 - Blastoise vs Alakazam
Vencedor: Alakazam
Combate 6 - Lapras vs Alakazam
Vencedor: Alakazam
Combate 7 - Snorlax vs Alakazam
Vencedor: Alakazam
EL VENCEDOR DE ESTA BATALLA ES: Usuario 2
Con su team de: Pidgeot, Alakazam, Rhydon, Arcanine, Exeggutor, Blastoise
```

ENTRADA 2

Usuario 1

```
1: Metapod
2: Metapod
3: Metapod
4: Metapod
5: Metapod
6: Metapod
```

Usuario 2

```
1: Metapod
2: Metapod
3: Metapod
4: Metapod
5: Metapod
6: Metapod
```

SALIDA 2

```
Combate 1 - Metapod vs Metapod
Vencedor: Metapod
Combate 2 - Metapod vs Metapod
Vencedor: Metapod
```

```
Combate 3 - Metapod vs Metapod
Vencedor: Metapod
Combate 4 - Metapod vs Metapod
Vencedor: Metapod
Combate 5 - Metapod vs Metapod
Vencedor: Metapod
Combate 6 - Metapod vs Metapod
Vencedor: Metapod
EL VENCEDOR DE ESTA BATALLA ES: Usuario 2
Con su team de: Metapod, Metapod, Metapod, Metapod, Metapod, Metapod
```

ENTRADA 3

Usuario 1

```
1: Venusaur
2: Lickitung
3: Farfetch'd
4: Aerodactyl
5: Parasect
6: Omanyte
```

Usuario 2

```
1: Poliwag
2: Poliwhirl
3: Golduck
4: Cubone
5: Exeggcute
6: Bulbasaur
```

SALIDA 3

```
Combate 1 - Venusaur vs Poliwag
Vencedor: Venusaur
Combate 2 - Venusaur vs Poliwhirl
Vencedor: Venusaur
Combate 3 - Venusaur vs Golduck
Vencedor: Venusaur
Combate 4 - Venusaur vs Cubone
Vencedor: Venusaur
Combate 5 - Venusaur vs Exeggcute
Vencedor: Exeggcute
Combate 6 - Lickitung vs Exeggcute
Vencedor: Exeggcute
Combate 7 - Farfetch'd vs Exeggcute
Vencedor: Farfetch'd
Combate 8 - Farfetch'd vs Bulbasaur
Vencedor: Farfetch'd
```

```
EL VENCEDOR DE ESTA BATALLA ES: Usuario 1
Con su team de: Venusaur, Lickitung, Farfetch'd, Aerodactyl, Parasect,
Omanyte
```