INF-253 Lenguajes de Programación
Tarea 1: Python

Profesor: José Luis Martí Lara, Roberto Diaz Urra
Ayudante Cátedras: Gabriela Acuña Benito, Hugo Sepúlveda Arriaza,
Lucio Fondón Rebolledo
Ayudante Tareas: Gabriel Carmona Tabja, Domingo Benoit Cea,
Héctor Larrañaga Jorquera, Ignacio Ulloa Huenul,
Joaquín Gatica Hernández, Javier Pérez Riveros,
José Runín Basáez, Rafael Aros Soto
Rodrigo Pérez Jamett

1 de septiembre de 2021

1. Un sueño increible

Un informático anómimo jugó demasiado un juego de Mario, luego de horas y horas moviéndose en el juego, el informático se cansó y se fue a dormir. Y ahí es donde comenzó algo increíble, se imaginó un lenguaje en el cual se podría manejar a base de movimientos de Mario. Lo más interesante es que a ese informático se imaginó todo el lenguaje, pero no sabía programarlo, por lo que le pide a ustedes que lo ayuden a desarrollar un intérprete para el lenguaje.

Se debe utilizar Python 3 y la librería RegEx para las expresiones regulares de la tarea, en caso de que alguna de estas dos condiciones no se cumpla no se revisará la tarea.

2. MovMarioLP

2.1. Acerca de MovMarioLP

La idea base del MovMarioLP es que según los movimientos que haga Mario se realice una acción. Este lenguaje además cuenta con una sección de memoria, que consiste en una matriz, que sirve para almacenar valores numéricos.

2.2. Matriz

La matriz consistirá de un tamaño de $n \times n$ (n siendo un valor que se especificará más tarde), donde inicialmente todos los valores parten en 0.

Además es una matriz cíclica, significa que si es que yo me muevo hacia arriba y estoy en límite, entonces iré a la última fila de la matriz, debido a que di la vuelta.

Se asume que todos los valores de la matriz son enteros.

2.3. Comandos

El lenguaje consiste de los siguientes comandos:

- $U[numero]$: se mueve una cantidad equivalente a número de posiciones hacia arriba dentro de la matriz.
- $D[numero]$: se mueve una cantidad equivalente a número de posiciones hacia abajo dentro de la matriz.
- $<[numero]$: se mueve una cantidad equivalente a número de posiciones hacia la izquierda dentro de la matriz.
- $>[numero]$: se mueve una cantidad equivalente a número de posiciones hacia la derecha dentro de la matriz.
- A: aumenta en uno el valor en donde se encuentra.
- B: disminuye en uno el valor en donde se encuentra.
- $X[dir]$: multiplica el valor en donde se encuentra por algún valor indicado por dir, donde dir indica cual es la casilla que se utiliza, usando las 4 direcciones definidas arriba.
- $Y[dir]$: divide el valor en donde se encuentra por algún valor indicado por dir, donde dir indica cual es la casilla que se utiliza, usando las 4 direcciones definidas arriba.
- $L[c|e]$: muestra el valor en donde se encuentra, donde si está acompañado de c se debe imprimir la versión caracter del entero y si está acompañado de e se debe imprimir la versión entero.
- R: reinicia el valor en donde se encuentra a 0.
- $?[dir][comando]$: pregunta si el valor en donde se consulta el valor que se encuentra en dir es mayor a 0 o no. Si es mayor, entonces realiza el comando, si no simplemente la ignora.
- Z: reinicia todos los valores de la matriz a 0.
- $S[c|e]$: muestra todos los valores dentro de la matriz (de izquierda a derecha, arriba hacia abajo), donde si está acompañado de c se debe imprimir la versión carácter del entero y si está acompañado de e se debe imprimir la versión entero.

Code 1: EBNF

```
condicional ::= '?' dir comando

dir ::= ('U' | 'D' | '<' | '>') numero {('U' | 'D' | '<' | '>') numero}

comando ::= condicional | operacion

operacion ::= dir | 'A' | 'B' | X dir | Y dir | 'L' tipo | 'R' | 'Z' | 'L' tipo

numero ::= 0 | no_zero {'0' | no_zero}

no_zero ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

tipo = 'c' | 'e'
```

2.4. Sobre algunos comandos

- La operación $Y[dir]$, si el valor el cual está en dir en 0 no se debe realizar la operación.
- La operación $L[c|e]$, si se encuentra Lc y el valor no está entre 32 y 127, entonces no se debe imprimir nada.
- La operación $S[c|e]$, si se encuentra Sc y el valor de alguna casilla y el valor no está entre 32 y 127, entonces no se debe imprimir ese valor.
- Notar que debido a la operación B pueden existir números negativos, por lo que si se decide imprimir por caracter, no se imprimirán.

2.5. Orden de operaciones

El orden de las operaciones consiste de izquierda a derecha. Además existen los paréntesis para darle orden a las operaciones que uno quiera realizar, donde el orden corresponde al par de paréntesis de más adentro y el de más la izquierda primero.

Por ejemplo:

Code 2: Ejemplo orden

```
(<1(>1)(A))
```

En este ejemplo la operación > 1 se realizará primero, luego A y finalmente < 1.

3. Su objetivo

Deben generar un programa el cual permita recibir una cantidad indefinida de líneas en un archivo **codigo.txt**, donde la primera línea siempre consistirá en el tamaño de la matriz que se utilizará y luego cada línea siguiente corresponderá a una secuencia de comandos.

Su programa debe actuar cómo un compilador y intérprete. Compilador: a partir del archivo código.txt, su programa debe generar un archivo .txt, llamado **errores.txt**, donde se encontrarán todas las líneas que tengan una sintaxis incorrecta, indicando junto a la línea el número de esta, en el caso que no hayan errores se deberá mostrar el archivo "No hay errores!". Intérprete: debe realizar la ejecución de todas las líneas correctas, en caso que no hayan líneas correctas deben mostrar por pantalla "No hay lineas correctas :c".

Siempre la posición inicial debe ser el $0, 0$ de la matriz.

3.1. Ejemplos

3.1.1. Ejemplo 1

Code 3: Ejemplo 1

```
13
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA>1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA>1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>1
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA>1
Sc
```

Code 4: errores.txt

```
No hay errores!
```

Code 5: consola

```
Hello
```

En este ejemplo la sintaxis está correcta, se observa que solo modifica 5 casillas de la matriz sumándole 1 al valor de distintas cantidades, luego imprime toda la matriz en forma de caracteres, como solo hay 5 casillas entre 32 y 127, se imprime solamente esas 5 casillas según la tabla ASCII, ya que su valor no es 127. Imprimiendo Hello.

3.1.2. Ejemplo 2

Code 6: Ejemplo 2

```
13
?A
>1>1>1>1ALe
```

Code 7: errores.txt

```
1 ?A
```

Code 8: consola

```
1
```

En ese ejemplo, la primera línea se encuentra incorrecta, ya que es un condicional que le falta *dir*. La segunda línea está correcta, por lo que se ejecuta y consta de solo 4 movimientos hacia la derecha, luego sumarle uno a esa casilla y luego imprimir ese valor como entero, siendo este entero 1.

Code 9: Ejemplo 3

```
4
AAD2
(?U2A(Le)Le)A
>1U1AAXD1<1
Se
(U2
D01
DU
```

Code 10: errores.txt

```
5 (U2
6 D01
7 DU
```

Code 11: consola

```
012000040020000000
```

En este ejemplo, hay 3 líneas incorrectas, el primera incorrecta se debe a que falta cerrar la parentesis, la segunda incorrecta se debe a que el número que acompaña a D comienza con 0, la tercera se debe a que los dos comandos de movimiento le faltan los números.

De las líneas correctas, la primera suma 2 al valor actual que es $0, 0$ y se mueve 2 para abajo. La segunda pregunta, primero imprime el valor de la casilla actual, ya que es el paréntesis más adentro y más a la izquierda, entonces imprime 0, luego pregunta si es que la casilla que está dos posiciones arriba es mayor a 0, como la respuesta es sí debido a que esa casilla vale 2, gracias al comando anterior, se realiza la operación A, al terminar eso se vuelve a imprimir la casilla actual, mostrando en pantalla 1, para finalizar la línea sumándole 1 a la casilla actual. La tercera línea primero se mueve una casilla a la derecha y uno hacia arriba, suma dos a la casilla actual, luego realiza la operación multiplicación en donde multiplica la casilla actual con la casilla que está uno abajo y uno a la izquierda, que en este caso vale 2, finalmente se imprime la matriz mostrando el resultado en pantalla.

4. Datos de vital importancia

- Para imprimir carácter se sigue la notación de la tabla ASCII, **EXCEPTO** por el valor 127 que en esta tarea se traduce por salto de línea.
- Todas las impresiones se deben realizar así tal cual como se indica.
- Si hay un abre paréntesis debe existir un cierra paréntesis, y viceversa.
- **Para toda la tarea debe hacer uso de expresiones regulares, si es que no usa no se revisará la tarea.**

5. Sobre Entrega

- Se debera entregar un programa llamado movmariolp.py.
- Si no existe orden en el código habrá descuento.
- Las funciones implementadas deben ser comentadas de la siguiente forma. **SE HARÁN DESCUENTOS POR FUNCIÓN NO COMENTADA**

  ”’

  Nombre de la función

  Parametro 1 : Tipo
  Parametro 2 : Tipo
  Parametro 3 : Tipo
  .....

  Breve descripción de lo que realiza la función y lo que retorna ”’
- Se debe trabajar de forma individual obligatoriamente.
- **La entrega debe realizarse en tar.gz y debe llevar el nombre: Tarea1LP_RolAlumno.tar.gz**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa.**
- La entrega será vía aula y el plazo máximo de entrega es hasta el **27 de septiembre a las 23:55 hora aula**.

- Por cada día de atraso se descontarán 20 puntos (10 puntos dentro la primera hora).
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

6. Calificación

6.1. Entrega

- Uso correcto de expresiones regulares (18 puntos)
- Detección de errores (17 puntos)
- Ejecución de comandos (65 puntos)

6.2. Descuentos

- Falta de comentarios (-10 puntos c/u MAX 30)
- Falta de README (-20 puntos)
- No respeta formato reglas de entrega (-30 puntos)