Instrucciones

1. PixelArt

Un jugador de Minecraft amante del pixelart, un día buscando creaciones de otros jugadores, se encontró con un sitio web que aseguraba tener diseños que nunca antes había visto, con instrucciones muy detalladas de cómo construirlos; sin embargo, ¡este sitio no contaba con imágenes para saber cómo se ven los resultados de seguir esas instrucciones!

Code 1: Ejemplo de instrucciones

```
Ancho 8
Color de fondo RGB(13,181,13)

    Avanzar Derecha Avanzar 2
    Pintar Negro Avanzar
    Repetir 2 veces { Pintar Negro Izquierda Avanzar }
    Pintar Negro
    Derecha Avanzar 3
    Pintar Negro Avanzar
    Repetir 2 veces { Pintar Negro Derecha Avanzar }
    Pintar Negro
    Izquierda Avanzar
    Repetir 3 veces { Avanzar Pintar Negro }
    Derecha Avanzar 3 Derecha
    Repetir 3 veces { Pintar Negro Avanzar }
    Derecha Avanzar
    Repetir 3 veces {
    Pintar Negro Avanzar
    Pintar Negro Derecha Avanzar
    Derecha Avanzar Derecha Derecha
    }
```

Es por lo anterior que este jugador le pidió a los alumnos de Lenguajes de Programación que utilicen los contenidos del curso para crear un programa capaz de interpretar las instrucciones del sitio y producir una imagen del resultado.

Para esta tarea se debe utilizar Python 3 y los siguientes paquetes:

- RegEx para las expresiones regulares de la tarea
- NumPy y Pillow para convertir los resultados a imágenes. El código para este proceso se les será entregado.

2. Lenguaje PixelArt (LPA)

2.1. Acerca de LPA

El Lenguaje PixelArt es un nuevo lenguaje interactivo que permite a un jugador, mediante una secuencia de instrucciones, producir uno de los pixelarts del sitio. Siempre se asume que el jugador comienza en la esquina superior izquierda apuntando hacia la derecha. El objetivo es simular el

procedimiento que seguirá un jugador y guardar el resultado como una matriz (lista de listas en Python) de valores RGB (tupla de 3 valores entre 0 y 255 en Python).

2.2. Matriz

La matriz sería de tamaño *n*×*n*, siendo *n* un valor que se especificará más adelante, donde inicialmente todos los valores parten en un mismo color.

2.3. Comandos

Todo archivo **siempre** comienza con los siguientes comandos en este orden:

- **Ancho** N: Indica que la imagen será de tamaño *N*×*N*.
- **Color de fondo** Color: Indica el color de fondo que tendrá la imagen, es decir, si un bloque no es pintado entonces será de este color. El formato que sigue *Color* se indica más adelante.

Posterior a estos habrá una línea en blanco y luego seguirán las instrucciones, las cuales pueden ser cualquier combinación de las siguientes:

- **Izquierda**: El jugador debe girar en 90° a la izquierda.
- **Derecha**: El jugador debe girar en 90° a la derecha.
- **Avanzar** N: El jugador debe avanzar *N* cuadros en la dirección en la que está mirando. En caso de que no se entregue un número, el jugador deber 'a avanzar 1 bloque.
- **Pintar** Color: El jugador debe pintar el bloque en el que se encuentra de color *Color*. El formato que sigue *Color* se indica más adelante.
- **Repetir** N **veces** {Instrucciones}: El jugador debe repetir las instrucciones indicadas en *Instrucciones N* veces. Las instrucciones pueden ser cualquiera de las anteriormente nombradas a excepción de **Ancho** y **Color de fondo**. Esto sígnica que pueden haber instrucciones de **Repetir** anidadas.

Cuando un comando requiera de un *Color* como parámetro, este podrá recibir uno de los siguientes valores:

- **Rojo**: Deberá pintar de color rojo, escribiendo en la matriz de memoria la tupla (255,0,0).
- **Verde**: Deberá pintar de color verde, escribiendo en la matriz de memoria la tupla (0,255,0).
- **Azul**: Deberá pintar de color azul, escribiendo en la matriz de memoria la tupla (0,0,255).
- **Negro**: Deberá pintar de color negro, escribiendo en la matriz de memoria la tupla (0,0,0).
- **Blanco**: Deberá pintar de color blanco, escribiendo en la matriz de memoria la tupla (255,255,255).
- **RGB(**R,G,B**)**: Deberá pintar del color indicado, escribiendo en la matriz de memoria la tupla (R,G,B).

A continuación se presenta un EBNF que describe formalmente la sintaxis que siguen las instrucciones de un programa:

Code 2: EBNF

```
                    instruccion ::= 'Izquierda' | 'Derecha'
                    | 'Avanzar' [numero] | 'Pintar' color
                    | 'Repetir' numero 'veces' '{' {instruccion} '}'

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

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

          color ::= 'Rojo' | 'Verde' | 'Azul' | 'Negro' | 'Blanco'
          | 'RGB(' numero ',' numero ',' numero ')'
```

Puede haber múltiples instrucciones en una línea. Adicionalmente las instrucciones de tipo Repetir pueden abarcar múltiples líneas.

2.4. Orden de operaciones

Las instrucciones siempre se ejecutan de izquierda a derecha.

3. Objetivo de la tarea

Se debe crear un programa el cual permita recibir una cantidad indefinida de líneas en un archivo codigo.txt, en donde la primera línea siempre indicara el tamaño de la matriz que se utilizará, la segunda el color de fondo, la tercera será una línea en blanco y luego cada línea siguiente corresponderá a una serie de instrucciones.

Su programa debe ser capaz interpretar el código, detectar errores de sintaxis, y detectar errores de ejecución. Sintaxis: a partir del archivo codigo.txt, su programa debe generar un archivo errores.txt, donde se encontraran todas las líneas que tengan una sintaxis incorrecta, indicando el número de línea en la cual se encuentra el error, en el caso que no hayan errores el archivo deberá contener la frase "No hay errores!". Interprete: debe producir pixelart.png y mostrar por consola los valores de la matriz RGB en caso de que la ejecución y la revisión de sintaxis sea exitosa. Si durante la ejecución las instrucciones causaran que el jugador salga del área que cubre la matriz, se debe indicar por consola la línea que produjo este error y terminar la ejecución.

3.1. Ejemplos

3.1.1. Ejemplo 1

Code 3: Ejemplo 1

```
Ancho 10
Color de fondo Blanco

Avanzar Derecha Avanzar
Repetir 4 veces {
Repetir 8 veces { Pintar Negro Avanzar } Derecha Derecha Avanzar Derecha
}
```

Code 4: errores.txt

```
No hay errores!
```

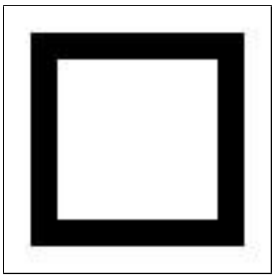

Figura 1: pixelart.png

3.1.2. Ejemplo 2

Code 5: Ejemplo 2

```
Ancho 3
Color de fondo RGB(0,0,0)

Pintar Rojo Avanzar 2 Derecha
Pintar Verde Avanzar 2 Derecha
Pintar Azul Avanzar 2 Derecha
Pintar Blanco Avanzar Derecha Avanzar
Pintar RGB(0,255,255)
```

Code 6: errores.txt

```
No hay errores!
```

Figura 2: pixelart.png

3.1.3. Ejemplo 3

Code 7: Ejemplo 3

```
Ancho 3
Color de fondo Negro

```

```
    Repetir 2 {
    Pintar RGB(255,0,0) Avanzar Derecha 1 Avanzar
    }
    Pintar RGB(255,0,0)
```

Code 8: errores.txt

```
4 Repetir 2 {
5 Pintar RGB(255,0,0) Avanzar Derecha 1 Avanzar
```