Teoría de Lenguajes
Primer Laboratorio – 2018

El propósito de este laboratorio es trabajar con expresiones regulares, para lo cual se propone escribir programas en el lenguaje Python, donde se ven ejemplos prácticos de uso de las mismas.

El foco es resolver los problemas haciendo un uso fuerte de las expresiones regulares, y no sustituyendolas por sentencias de programación.

Deben usarse expresiones regulares lo más genéricas posible, no dependientes de datos de entrada particulares. En general los programas se pueden resolver en menos de 10 líneas.

Modo de Trabajo

- Se indicarán 5 programas a realizar.

- Se entregará para cada programa: un archivo de entrada y un archivo con la salida “oficial” que contiene la salida que se debería obtener para el archivo de entrada correspondiente.

- El estudiante en base a lo especificado por la letra, la entrada y la correspondiente salida oficial deberá realizar su programa.

- La salida del programa del estudiante deberá ser idéntica a la salida oficial. Algunos detalles de cómo debe trabajar su programa surgen de analizar la entrada y su correspondiente salida. Para las comparaciones se entregará el programa “diff.exe” para Windows que tomando como entrada dos archivos detecta si hay diferencias.

- Se deberá respetar los nombres de los archivos de entrada, de salida, y de los programas Python.

- Se proporcionará un archivo comprimido que contiene lo siguiente:
    - los archivos de entrada
    - los archivos con las salidas "oficiales"
    - los archivos Python que deben implementar, dentro del subdirectorio: programas
    - scripts para facilitar las tareas de ejecutar y comparar las salidas
    - el programa diff.exe para comparar las salidas

programa1.py

Dado un subprograma en pascal, despliega todas las condiciones de cada sentencia de control: *if*, *for* y *while* de cada sentencia del programa.

Ejemplo de entrada:

```
procedure BubbleSort(numbers : array of integer; size : integer);
var
      i, j, temp : Integer;

begin
      for i := size-1 downto 1 do
         for j := 2 to i do
            if (numbers[j-1] > numbers[j]) then
            begin
                  temp := numbers[j-1];
                  numbers[j-1] := numbers[j];
                  numbers[j] := temp;
            end;
end;
```

Ejemplo de salida :

```
if:
(numbers[j-1] > numbers[j])
for:
size-1 downto 1
2 to i
```

programa2.py

Dado un subprograma en pascal despliega un resumen con la cantidad de sentencias de control: *if*, *for*, *while* y *repeat* que contiene.

Ejemplo de entrada :

```
procedure QSort(numbers : array of integer; left : integer; right : integer);
var
	pivot, l_ptr, r_ptr : Integer;
begin
	l_ptr := left;
	r_ptr := right;
	pivot := numbers[left];

	while (left < right) do
	begin
		while ((numbers[right] >= pivot) and (left < right)) do
			right := right - 1;

		if (left <> right) then
		begin
			numbers[left] := numbers[right];
			left := left + 1;
		end;

		while ((numbers[left] <= pivot) and (left < right)) do
			left := left + 1;

		if (left <> right) then
		begin
			numbers[right] := numbers[left];
			right := right - 1;
		end;
	end;
	numbers[left] := pivot;
	pivot := left;
	left := l_ptr;
	right := r_ptr;
	if (left < pivot) then
		QSort(numbers, left, pivot-1);
	if (right > pivot) then
		QSort(numbers, pivot+1, right);
end;
```

Ejemplo de salida:

```
if 4
for 0
while 3
repeat 0
```

programa3.py

Dado un subprograma en pascal, para cada línea del programa que contiene un := (símbolo de asignación), devuelve el texto que está a la izquierda del := sin contar los espacios, tabuladores, pegados al comienzo de la línea.

Ejemplo de entrada:

```
procedure InsertionSort(numbers : array of integer; size : integer);
var
        i, j, index : integer;

begin
        for i := 2 to size-1 do
        begin
                index := numbers[i];  j := i;
                while ((j > 1) AND (numbers[j-1] > index)) do
                begin
                        numbers[j] := numbers[j-1];
                        j := j - 1;
                end;
                numbers[j] := index;
        end;
end.
```

Ejemplo de salida:

```
for i
index
numbers[j]
j
numbers[j]
```

programa4.py

Dado un subprograma en pascal, retorna el mismo programa sin comentarios.

Ejemplo de entrada:

```
procedure Triangulo();
    { Programa de ejemplo:
      para calcular el área de un triángulo. }
   var altura,base,area: real;
   begin // Inicio
     (* ingresar datos *)
     write('Ingrese altura y base: ');
     readLn(altura,base);

     (* calcular resultado *)
     area := base * altura / 2;

     (* mostrar resultado *)
     writeLn('El área del triangulo es: ',area:7:2);
end; // Fin
```

Ejemplo de salida:

```
procedure Triangulo();

var altura,base,area: real;
begin

  write('Ingrese altura y base: ');
  readLn(altura,base);


  area := base * altura / 2;


  writeLn('El área del triangulo es: ',area:7:2);
end;
```

programa5.py

Dado un subprograma en pascal, devolver el conjunto de duplas “(var, n)” donde var es el nombre de una variable declarada dentro del procedimiento y n la cantidad de ocurrencias de dicha variable en el procedimiento. Se asume que no hay subprogramas andiados

Ejemplo de entrada:

```
Procedure QSort(numbers : Array of Integer; left : Integer; right : Integer);
Var  pivot, l_ptr, r_ptr : Integer;
Begin
	l_ptr := left;
	r_ptr := right;
	pivot := numbers[left];

	While (left < right) do
	Begin
		While ((numbers[right] >= pivot) AND (left < right)) do
			right := right - 1;

		If (left <> right) Then
		Begin
			numbers[left] := numbers[right];
			left := left + 1;
		End;

		While ((numbers[left] <= pivot) AND (left < right)) do
			left := left + 1;

		If (left <> right) Then
		Begin
			numbers[right] := numbers[left];
			right := right - 1;
		End;
	End;

	numbers[left] := pivot;
	pivot := left;
	left := l_ptr;
	right := r_ptr;

	If (left < pivot) Then
		QSort(numbers, left, pivot-1);

	If (right > pivot) Then
		QSort(numbers, pivot+1, right);
End;
```

Ejemplo salida:

```
(pivot,9)
(l_ptr,2)
(r_ptr,2)
```

Herramientas a utilizar

Se usará Anaconda de Python 3.6. Por más información ver instructivo.

Desarrollo del trabajo

Los trabajos se deben realizar en grupos de 2, 3 o 4 estudiantes. No puede realizarse en forma individual. Se sancionará con la pérdida del laboratorio si se detectan trabajos copiados, o si se hace una mala utilización de los foros de consulta.

Entrega

Se realizará vía web, y se comunicarán los detalles oportunamente. La misma se habilitará un par de días antes del vencimiento el día **24 de abril hasta las 23:55 hs.**

Se entregarán los **5 programas**, cuyos nombres deberán ser exactamente los especificados (respetando mayúsculas, minúsculas, guiones, etc.).
Se entregará un archivo de texto de nombre **integrantes.txt** que contendrá la CI (7 dígitos) y el nombre de cada uno de los integrantes del grupo (sin usar tildes), y a continuación algún comentario que quieran hacer en el caso que algún programa no les hubiera funcionado correctamente. No necesitan inscribir el grupo previamente.
Formato del archivo integrantes.txt:

```
3123456,Gonzalez,Pablo
4567123,Martinez,Veronica
3444555,Garcia Rodriguez,Juan Jose
Comentarios (sólo si algún programa no hubiera funcionado bien).
```

Se debe respetar el formato del archivo:

- datos de los integrantes: “CI (7 dígitos sin guiones)” “coma” “Apellido/s” “coma” “Nombre/s”
- un integrante por línea, en las primeras líneas del archivo.

Corrección

Son parte de las normas usar los scripts entregados y los nombres de los programas indicados. Es importante cumplir la norma y se penalizará en la corrección si no se sigue adecuadamente.
La corrección se realizará usando los mismos scripts entregados para probar, por lo tanto si la ejecución cancela, o hay diferencia en los archivos de salida con la solución oficial, la solución no se considerará correcta.
Se usarán además otros tests de prueba en la corrección, para testear posibles errores que se pudieran cometer.
Es importante, como ya se aclaró al principio, usar la funcionalidad de las expresiones regulares del lenguaje, y no sustituirlas por sentencias de programación.

Observaciones sobre los códigos Pascal de entrada

Todos los códigos son correctos y compilan.