Facultad de Ciencias Sociales y Administrativas

Licenciatura en Informática y Desarrollo de Software
Licenciatura en Telecomunicaciones

PRÁCTICO DE ENSEÑANZA
ASIGNATURA: Programación I PRÁCTICO Nº: 4 FECHA DE FINALIZACIÓN: 17 de Mayo de 2021 TEMA: Estructuras de decisión DOCENTES RESPONSABLES: Eraso Martín, Ontivero Renzo
NOMBRE Y APELLIDO DE LOS ALUMNOS: CURSO Y COMISIÓN:
OBJETIVO: Resolver correctamente ejercicios básicos de pseudocódigo utilizando PSeInt. FORMA DE APROBACIÓN: Que resuelva correctamente al menos 3 de los 5 ejercicios propuestos. Que entregue los trabajos por el campus virtual y github.

OBSERVACIONES

- **El trabajo práctico deberá ser resuelto en grupos de 2 a 3 integrantes**
- Los algoritmos deberán ser resueltos con el software **PSeInt**
- Utilizar la siguiente nomenclatura para nombrar el código fuente de los ejercicios: TP4_EJX.psc, donde la letra X es el número de ejercicio.
- La entrega del trabajo práctico deberá realizarse por el campus virtual y deberá ser un archivo en formato **.zip** que contenga todos los códigos correspondiente a los ejercicios. El nombre del documento será **APELLIDOS-INTEGRANTES.zip**
- Además se deberán subir los ejercicios a **github** según los pasos vistos en clase de práctica. No se aceptarán trabajos subidos por medio de la aplicación web (Github). Deberán utilizar git por consola para realizar la entrega.

CONSIGNAS

1. Desarrollar un algoritmo que permita almacenar en un arreglo los cien primeros números pares.
   Luego recorrer el arreglo y mostrar los números que contiene el arreglo.

2. Desarrollar un algoritmo que llene un arreglo con 20 números de forma aleatoria entre 0 y 100 . Luego busque en el arreglo, un número generado de forma aleatoria entre 0 y 100, y finalmente muestre: a)- El número a buscar, b)- El arreglo, c)- Si encontró el número, la posición donde fue hallado, o d)- Si no encontró el número.

3. Desarrollar un algoritmo que permita obtener los **números primos** del rango de valores del 1 al 100. Cuando encuentre un número primo, lo deberá almacenar en un arreglo.
   Finalmente se deberá recorrer el arreglo, mostrar los números primos encontrados y en qué posición del arreglo se encuentra.
   Como ayuda se adjunta este algoritmo que se encarga de determinar si un número ingresado por el usuario es primo o no.

```
Algoritmo primos
	Escribir "Ingrese número para analizar si es primo: "
	// Guardamos el número a analizar en la variable n.
	Leer n
	// Creamos una variable para contar cuantas veces puede ser dividido n.
	contar <- 0
	// Recorremos en un bucle todas las posibilidades desde 1 hasta el número ingresado.
	Para i<-1 Hasta n Hacer
		// Dividimos el número por la variable del bucle i.
		Si n%i=0 Entonces
			// Si el resto da 0 sumamos 1 a la variable contar.
			contar <- contar+1
		FinSi
	FinPara
	// Si contar sumo 2 el número es primo.
	Si contar=2 Entonces
		Escribir n," es primo."
	SiNo
		Escribir n," no es primo."
	FinSi
FinAlgoritmo
```

4. Desarrollar un algoritmo que permita generar una contraseña aleatoria numérica de 8 dígitos. Para ello se deberá utilizar un arreglo de tamaño 8, que permita almacenar en cada una de sus posiciones los dígitos obtenidos de forma aleatoria.

Finalmente se deberá recorrer nuevamente el arreglo para mostrar la contraseña obtenida. Para mostrar la contraseña en una instrucción **Escribir**, investigar acerca de las funciones **concatenar()** y **convertiratexto()**.

```
Algoritmo generador2
	Escribir "Ingresar la longitud de la contraseña:"
	Leer L
	code <- "" // Iniciamos una variable con una cadena vacía.
	Escribir 'Generando contraseña numérica...'
	Para i<-1 Hasta L Hacer
		// Concatenamos números aleatorios de 0 al 9 con la longitud re-
querida.
		code <- concatenar(code,(convertiratexto(aleatorio(0,9))))
	FinPara
	// Mostramos el código resultante de la concatenación guardada en la
variable code.
	Escribir code
FinAlgoritmo
```

5. Desarrolle un algoritmo que permita jugar al TA-TE-TI contra la computadora.

Reglas de juego:

- Se deben permitir como máximo 3 jugadas por cada participante.
- El usuario deberá seleccionar una posición del TA TE TI indicando número de Fila y Columna. Comprobar si está utilizado por otro jugador y avisar al usuario que seleccione otra.
- Luego de cada movimiento se deberá mostrar el estado actual del TATETI
- La computadora deberá seleccionar posiciones de forma aleatoria.

También se deberá validar si la posición está ocupada.

- Al finalizar los 6 movimientos, calcular si hay ganador o empate. Mostrar los resultados.

Condiciones:

- Utilizar arreglos para resolver este problema
- Puede utilizar un menú de opciones para indicar al usuario la posición en la que desea colocar su ficha
- Validar posibles errores en las jugadas del usuario y computadora
- **Pueden utilizar como variante desarrollar el juego para dos jugadores humanos o dos jugadores computadora**.

No olvidar

- Definición e inicialización de variables
- Indentado de código
- Convención para nombre de variables/CONSTANTES camelCase o sanake_case
- Comentarios de bloques de código
- Uso correcto de estructuras de decisión y repetitivas
- Refactorizar código repetitivo para optimizar el algoritmo