Programación II

Práctica Calificada 1

Pregrado

2023-2

Profesor: Edson Mendiolaza

Lab. 2.05

Indicaciones específicas:

- Esta evaluación contiene 8 páginas (incluyendo esta página) con 3 preguntas. El total de puntos son 20.
- El tiempo límite para la evaluación es 100 minutos.
- Está prohibido el uso de celulares
- Crea una carpeta llamada: PC1
- Luego crea una subcarpeta para cada pregunta:
  1. p1
  2. p2
  3. p3
- Una vez que termines de realizar las 3 preguntas:
  1. Elimina el directorio cMake-Buil-debug y el directorio .idea de cada pregunta
  2. Comprime la carpeta PC1 usando el winzip y obtendras el archivo PC1.zip
- Sube el archivo **PC1.ZIP** al gradescoupe a www.gradescope.com.
- Recuerda que solo se calificará si has enviado en el formato indicado.

Competencias:

- Para los alumnos de la carrera de Ciencia de la Computación

  Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (Evaluar)

  Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución.(Usar)

  Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (Usar)

- Para los alumnos de las carreras de Ingeniería

  Capacidad de aplicar conocimientos de matemáticas (nivel 3)

  Capacidad de aplicar conocimientos de ingeniería(nivel 2)

  Capacidad para diseñar un sistema, un componente o un proceso para satisfacer las necesidades deseadas dentro de restricciones realistas (nivel 2)

Calificación:

Tabla de puntos (sólo para uso del professor)

Question	Points	Score
1	7	
2	7	
3	6	
Total:	20	

1. (7 points) Evalúa estructuras de control

La **encriptación** es el conjunto de procedimientos, métodos y enfoques para proteger los datos confidenciales del acceso de terceros y realizar comunicaciones. Por ello, el departamento de Seguridad Informática necesita encriptar los códigos de "N" empleados. Para ello le solicitan que escriba un programa que genere aleatoriamente el código de cada empleado. Este código consiste en un **número de 8 dígitos.**

Por ejemplo,
Si se genera el código de 8 dígitos: **14508592**
El código encriptado sería: **HIT WOW**

Nota: Puede ingresar manualmente los datos de entrada o generarlos aleatoriamente.

Para realizar la encriptación, **debe utilizar operaciones matemáticas para dividir el número en dos partes** como se muestra a continuación:

1 4 5 0 — 1er número resultante

8 5 9 2 — 2do número resultante

Luego, cada número resultante debe encriptarse según la siguiente regla:

Primer número resultante	
Regla	**Clave de encriptación**
>= 1000 y <= 4000	Letra 'HIT'
> 4000 y <= 9000	Letra 'XXL'
En otros casos	'#@&'

Segundo número resultante	
Regla	**Clave de encriptación**
>= 1000 y <= 4000	Letra 'TAO'
> 4000 y <= 9000	Letra 'WOW'
En otros casos	'&*@'

La cantidad "N" de empelados puede ser un valor ingresado por teclado o un valor aleatorio generado por el computador. Este valor N debe estar validado en el rango de 100<=N<=1000;

Para el número aleatorio utilizar la siguiente semilla:
srand(time(nullptr))
Fórmula para el número aleatorio:
limite_inferior + rand() % (limite_superior +1 - limite_inferior)

Se muestra a continuación un ejemplo del funcionamiento del programa.

```
Empleado 1
----------
Codigo: 14508592
Codigo encriptado: HIT WOW

Empleado 2
----------
Codigo: 60019001
Codigo encriptado: XXL &*@
```

```
Empleado 3
----------
Codigo: 33304500
Codigo encriptado: HIT WOW

Empleado N
----------
Codigo:
Codigo encriptado:
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (3pts)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (2pts)	La ejecución es correcta (1pts)	La ejecución no es correcta (0.5pts)
Sintaxis	No existen errores sintácticos o de compilación (2pts)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (1.5pts)	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (1pts).	El código tiene errores de sintaxis que afectan el resultado (0.5pts).
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (2pts)	El código es de buen performance durante la ejecución (1.5pts).	El código no está optimizado pero la ejecución no es deficiente(1pts).	El código no está optimizado y la ejecución es deficiente (0pts).

2. (7 points) Evalúa Funciones

Elabore un programa que, haciendo uso de funciones, simule el juego de adivinar un número generado por el computador. El programa consiste en solicitar al computador que genere un numero aleatorio en el siguiente intervalo **1 <= numero_aleatorio <= 100.** Luego, el usuario tiene 3 intentos para adivinar el número generado. Por cada intento, **el programa debe dar una pista al usuario,** que indique si el número que ingresó es mayor o menor al número generado. Ver ejemplo

El juego termina cuando el usuario lo indique. Es decir, el juego se repite indefinidamente hasta que el usuario decida que ya no desea continuar. Cuando el juego termine, se debe mostrar el siguiente **Reporte**:

- ✓ Nro. total de jugadas realizadas
- ✓ Nro. de veces que adivinó el número en el primer intento

Tener en cuenta que:

- ✓ Si el usuario adivina en el primer intento, ya no debe pedir el segundo ni tercer intento.
- ✓ Si el usuario adivina en el segundo intento, ya no debe pedir el tercer intento.

En la solución es obligatorio utilizar:

- ✓ Una función para generar el numero aleatorio
- ✓ Una función para mostrar las pistas al usuario
- ✓ Una función parar mostrar el reporte

Se muestra a continuación un **ejemplo** del funcionamiento del programa.

```
                JUEGO
          ADIVINA EL NUMERO

 Ingrese Intento 1:9
 Tu numero es menor al numero generado!!!

 Ingrese Intento 2:19
 Tu numero es menor al numero generado!!!

 Ingrese Intento 3:90
 Tu numero es mayor al numero generado!!!
         Desea jugar otra vez [S]si [N]no ->S


 Ingrese Intento 1:60
 Tu numero es mayor al numero generado!!!

 Ingrese Intento 2:90
 Tu numero es mayor al numero generado!!!

 Ingrese Intento 3:15
 Tu numero es menor al numero generado!!!

         Desea jugar otra vez [S]si [N]no ->
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (3pts)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (2pts)	La ejecución es correcta (1pts)	La ejecución no es correcta (0.5pts)
Sintaxis	No existen errores sintácticos o de compilación (2pts)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (1.5pts)	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (1pts).	El código tiene errores de sintaxis que afectan el resultado (0.5pts).
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (2pts)	El código es de buen performance durante la ejecución (1.5pts).	El código no está optimizado pero la ejecución no es deficiente(1pts).	El código no está optimizado y la ejecución es deficiente (0pts).

3. (6 points) **Evalúa Punteros y recursividad**

Elabore un programa que, haciendo uso de punteros y recursividad, resuelva cada una de las opciones descritas en el siguiente menú.

MENU DE OPCIONES

1. Invertir un número ingresado y mostrarlo en pantalla
2. Hallar la suma de dígitos de un numero ingresado y mostrarlo en pantalla.
3. Convertir a binario un numero ingresado y mostrar el numero binario en pantalla
4. Salir

En la solución es obligatorio que:

- ✓ El rango del número ingresado para todas las opciones debe estar en el intervalo: 100 <=numero<=1 000 000
- ✓ En la opción **1** debe utilizar una función recursiva.
- ✓ En la opción **2** del menú debe utilizar una **función repetitiva con parámetros punteros**
- ✓ En la opción **3** del menú debe utilizar una **función recursiva con parámetros punteros**

Se muestra a continuación un ejemplo del funcionamiento del programa.

```
 MENU
1. Invertir y mostrar un numero
2. Hallar la suma de digitos
3. Convertir a binario
4. Salir
Elija una opcion :1

Ingrese el 100=<numero<=1000000 :4569
       NUMERO INVERTIDO = 9654
..presione una tecla para continuar...

 MENU
1. Invertir y mostrar un numero
2. Hallar la suma de digitos
3. Convertir a binario
4. Salir
Elija una opcion :2
Ingrese el 100=<numero<=1000000 :77889
        SUMA DE SUS DIGITOS:39
..presione una tecla para continuar...
```

```
 MENU
1. Invertir y mostrar un numero
2. Hallar la suma de digitos
3. Convertir a binario
4. Salir
Elija una opcion :3

Ingrese el 100=<numero<=1000000 :4569
         EL NUMERO EN BINARIO = 1000111011001
..presione una tecla para continuar...


 MENU
1. Invertir y mostrar un numero
2. Hallar la suma de digitos
3. Convertir a binario
4. Salir
Elija una opcion :4

Process finished with exit code 0
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (2pts)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (2pts)	La ejecución es correcta (1pts)	La ejecución no es correcta (0.5pts)
Sintaxis	No existen errores sintácticos o de compilación (2pts)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (1.5pts)	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (1pts).	El código tiene errores de sintaxis que afectan el resultado (0.5pts).
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (2pts)	El código es de buen performance durante la ejecución (1.5pts).	El código no está optimizado pero la ejecución no es deficiente(1pts).	El código no está optimizado y la ejecución es deficiente (0pts).