UNIVERSIDAD NEBRIJA

ALUMNO:

EPS -Ingeniería Informática

Asignatura: G0460006 Programación I – TURNO 1

Curso: 2022/2023 Examen: Final Fecha: 26-6-2023 (8:00-11:00)
Semestre: 1º Convocatoria: Extraordinaria

Parte Práctica (10 puntos; 70% nota final) - Tiempo: 2h40m

Los ficheros correspondientes a los ejercicios se deben entregar en la actividad correspondiente a través del campus antes de la finalización del tiempo establecido. Cada fichero se llamará EjercicioX, donde la X será el número de dicho ejercicio. Los únicos formatos válidos serán **txt** o **cpp**, siempre **sin comprimir**.

Criterios generales de evaluación

Funciones/Métodos: Si no se usa el paso por referencia constante cuando las variables de los parámetros de entrada no son de tipo simple.	40%
Tipos de datos y variables: • Uso de variables globales (fuera del ámbito de una función). • Si no se usan los tipos contenedor vistos en clase (std::array; std::vector; std::set; std::string, etc.) para las variables que lo necesiten. • Si no se usan punteros inteligentes (std::unique_ptr; std::shared_ptr) cuando sea necesario	 0% 0% 0%
El **programa no compila** o **no se asemeja** a lo pedido**.**	0%
Si no se cumplen los **criterios de entrega** indicados en la **actividad/examen.**	0%

Criterios particulares de evaluación

El elemento evaluable no compila o no se asemeja a lo que se pide	0%
El elemento evaluable no se aproxima suficientemente a lo pedido	40%
El elemento evaluable se aproxima suficientemente a lo pedido	60%
El elemento evaluable funciona correctamente y las estrategias y elementos de código elegidos son adecuados.	100%

IMPORTANTE:

- Todos los ejercicios del examen deberán ser resueltos de forma **algorítmica**, es decir, la **solución** propuesta tendrá que ser **general** y **no particular** para unos determinados datos/valores.
- Todos los ejercicios resueltos sin utilizar funciones cuando sea apropiado se valorarán con una nota máxima del 60% sobre la calificación prevista.
- Se recomienda una primera lectura del examen completo para planificar la realización del examen. Y una segunda lectura detallada antes de la realización de cada uno de los ejercicios propuestos.

UNIVERSIDAD NEBRIJA

Ejercicio 1 (3 puntos)

Escriba un programa en C++11, ***Ejercicio1***, utilizando los tipos de datos, las **estructuras de control** y las **funciones** necesarias para solicitar una **matriz cuadrada** al usuario y calcular su **traspuesta**. Después, hay que determinar si la matriz original es ***simétrica, antisimétrica*** o ***nada de lo anterior***.

La ***matriz traspuesta,*** A^t, de una matriz A es la **matriz** que se obtiene de **sustituir las filas por columnas** (o las columnas por filas).

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad A^t = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

Una matriz cuadrada es **simétrica** si coincide con su traspuesta, es decir, $A = A^t$.

$$A = A^t = \begin{pmatrix} 1 & 4 & 7 \\ 4 & 5 & 8 \\ 7 & 8 & 9 \end{pmatrix}$$

Una matriz cuadrada es **antisimétrica** si su traspuesta coincide con la matriz cambiada de signo, es decir, $A = -A^t$. Por tanto, los valores de su diagonal principal son **todos iguales a 0.**

$$A = \begin{pmatrix} 0 & 1 & -8 \\ -1 & 0 & 6 \\ 8 & -6 & 0 \end{pmatrix} \quad A^t = \begin{pmatrix} 0 & 1 & -8 \\ -1 & 0 & 6 \\ 8 & -6 & 0 \end{pmatrix}$$

El proceso debe realizar las siguientes tareas:

- Solicitar al usuario números enteros para crear una **matriz cuadrada** de dimensión definida a través del **valor de una constante declarada** en la función ***main*** del programa **[0,5 puntos]**.
- Calcular la **matriz traspuesta** de la matriz dato **[0,5 puntos]**.
- Determinar si la matriz original es **simétrica**, **antisimétrica** o **nada [0,75 puntos]**.
- Mostrar por pantalla el contenido de la **matriz cuadrada original**, de su **traspuesta** e indicar si es una matriz **simétrica**, **antisimétrica** o **nada [0,75 puntos]**.
- Finalmente, se preguntará al usuario si desea **repetir el proceso** con una nueva matriz o finalizar la ejecución **[0,5 puntos]**.

UNIVERSIDAD NEBRIJA

Ejercicio 2 (3 puntos)

Escriba un programa en C++11, ***Ejercicio2***, utilizando los tipos de datos, las **estructuras de control** y las **funciones** necesarias para determinar las **parejas de números primos gemelos** existentes en el **intervalo [100, 200]**, valores mayores e iguales a 100 y menores e iguales a 200. Las parejas de números primos gemelos del intervalo [100, 200] se deben almacenar en un **contenedor de tipo set (std::set)**.

Se dice que dos números primos, p y q, son **primos gemelos** si, siendo **p < q**, se cumple que: **q – p = 2**. Además, se sabe que:

- Todos los **números primos**, excepto el 2, son **números impares**.
- Los únicos **números primos consecutivos** son el 2 y el 3.

Por ejemplo, algunas **parejas de números primos gemelos** son: (3, 5), (5, 7), (11, 13),…, (239, 241), (269, 271), etc.

El proceso debe realizar las siguientes tareas:

- Determinar si un par de números dentro del intervalo [100, 200] son **primos gemelos** realizando las siguientes operaciones a través de funciones:
  - Determinar **si un número es primo o no [1,25 puntos]**.
  - Comprobar **si dos números son primos gemelos e incluirlos en un conjunto [0,5 puntos]**.
  - Incluir las **parejas de números primos gemelos** existentes en el intervalo establecido dentro del contenedor indicado **[0,25 puntos]**.
- Mostrar por pantalla la relación de parejas de números primos gemelos **[1 punto]**.

UNIVERSIDAD NEBRIJA

Ejercicio 3 (4 puntos)

Escriba un programa en C++11, ***Ejercicio3***, utilizando los tipos de datos, las **estructuras de control** y las **funciones** necesarias para capturar de forma cíclica, hasta que el usuario desee finalizar el programa, una sucesión de longitud indeterminada de caracteres 0 y 1 obtenidos por teclado:

- Implementar el código necesario para adquirir la información del usuario desde el teclado teniendo en cuenta los siguientes requisitos: **[0,75 puntos]**
    - El carácter punto detendrá la captura de información
    - Cualquier carácter diferente a 0, 1 o '.' será ignorado
- Implementar una función que a partir de una variable de tipo contenedor que almacene la cadena de caracteres **muestre por el terminal los valores almacenados [0,25 puntos]**
- Implementar una función que a partir de una variable de tipo contenedor que almacena la cadena de caracteres adquirida desde el teclado y una variable de tipo entero: **[1,5 puntos]**
    - Devuelva el **máximo número de valores consecutivos indicado por la variable de tipo entero**. Por ejemplo:
        - Si la variable de tipo entero contuviera un valor entero diferente a 0 o 1 la función devolverá el valor -1
        - Si la secuencia fuera 00011110 y la variable entera 1 el retorno sería 4
        - Si la secuencia fuera 00011110 y la variable entera 0 el retorno sería 3
        - Si la secuencia fuera 00001111 y la variable entera 1 el retorno sería 4
- Implementar una función de etiqueta ***rotar*** que a partir de una variable de tipo contenedor que almacena la cadena de caracteres adquirida desde el teclado y una variable de tipo bool se encargue de: **[1,5 puntos]**
    - Devuelver la sucesión de ceros y unos **rotada una posición a la derecha** si la variable de tipo bool es **true**, **a la izquierda si es false**.
        - Por ejemplo, si la secuencia fuera 01110011
            - Y el booleano fuera **false** la secuencia que debería devolver la función ***rotar*** sería 11100110
            - Y el booleano fuera **true** 10111001

Para puntuar los diferentes apartados de este ejercicio debes probarlos en el main.

No se tendrá en cuenta para la puntuación de este ejercicio los apartados no probados.