Tarea #4. Lenguajes de programación.

NOTAS:

A) Las aplicaciones deben usar exclusivamente los conceptos aprendidos.

B) Se debe crear un repositorio con el nombre tarea4

C) Se deben crear subcarpetas para cada uno de los puntos, con nombres: punto1, punto2, punto3, … etc. Donde cada carpeta contiene los archivos fuente ".cpp" y de cabecera ".hpp"

Las siguientes aplicaciones DEBEN incluir el uso de arreglos dinámicos, estructuras y deben estar implementadas en múltiples archivos.

1. Elementos únicos en un conjunto.
   Construir una aplicación con las siguientes funcionalidades:
   - Recibe un conjunto de números enteros ingresados por el usuario y les guarda dentro de un arreglo dinámico.
   - Calcula el conjunto de valores únicos dentro del conjunto original.
   - Imprime a consola ambos conjuntos de números enteros.

   EJO: Si el usuario ingresa [10,1,2,5,2,5,5,5,3,9,3,3], el conjunto de valores únicos es [10,1,2,5,3,9].

2. Histograma de un conjunto de enteros.
   Construir una aplicación para imprimir a consola el histograma de un conjunto de enteros.
   - Construya la aplicación partiendo del punto anterior.
   - Incluya una función que pueda ordenar los elementos de manera ascendente. (AYUDA: revise el reto de clase para el algoritmo "bubble").
   - Empleando el conjunto de números únicos ordenado, imprima a consola un histograma.

   EJO: si el usuario ingresa [10,1,2,5,2,5,5,5,3,9,3,3], el histograma debe tener la apariencia:

   1 *

   2 **

   3 ***

   5 ****

   9 *

   10 *

3. Revise el punto 5 de la tarea #3, repita el ejercicio con una división el código en múltiples archivos. Modifique el código para que el usuario pueda ingresar arreglos de cualquier tamaño.

4. Codificación de una señal.
   Queremos tener una aplicación para imprimir en consola un mensaje que solo se pueda leer a través de una clave. El proceso de generar el mensaje se llama codificación y la lectura se denomina decodificación. La codificación consiste en "desplazar" cada letra de la frase, una cantidad determinada de posiciones en al abecedario; por ejemplo si el desplazamiento es tres (3), la letra "a" se convierte en "d", la letra "b" se convierte en "e" y así sucesivamente. Al finalizar el abecedario el desplazamiento comienza de nuevo al inicio, de tal manera que la letra "x" se convierte "a", la letra "y" se convierte en "b" y la letra "z" se convierte en "c". Un resultado del mensaje codificado a tres caracteres es:

   "**hello world**" ➔ "**khoor zruog**"

   La aplicación debe recibir una frase del usuario y el número de caracteres para el desplazamiento. La aplicación debe codificar la frase e igualmente debe mostrar la decodificación, imprimiendo el mensaje original de nuevo.
   Nota: utilice el abecedario en inglés para aprovechar la conversión de la frase a ASCII.

5. Temporizador.
   Queremos crear una aplicación que despliegue un temporizador en consola. El usuario debe ingresar horas, minutos y segundos de inicio. La consola debe imprimir el conteo decreciente y desplegar un mensaje de alerta al final. El conteo debe hacerse a través de una estructura con elementos para el número de horas, minutos y segundos, de tal manera que en consola se imprima algo como:

   01 : 30 : 58

   Donde el conteo se encuentra en 01 horas, 30 minutos, 58 segundos.

   Como ayuda considere el siguiente código que despliega un conteo de 00 a 60 segundos en la consola.

```cpp
#include <iostream>
#include <iomanip>          //Input Output manipulation library
#include <windows.h>        //Windows utilities library

using namespace std;

int main()
{
    for (int i = 0; i < 60; i++){
            Sleep(1000);            //delay execution by 1000milisecons
            system("cls");          //Clean console
            cout << setfill('0') << setw(2) << i << endl;       //Print interger numbers with two digits (00, 01, 02, ...)
        }
}
```