UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II

Práctica Calificada 1 Extemporánea
Pregrado
2024-0
Profesor: Ian Paul Brossard
Lab 1.01

Indicaciones específicas:

- Esta evaluación contiene 6 páginas (incluyendo esta página) con 3 preguntas. El total de puntos es 20.
- El tiempo límite para la evaluación es 100 minutos.
- Cada pregunta deberá ser respondida en un archivo diferente. Se deberá usar los siguientes nombres de archivos:
  - p1.cpp
  - p2.cpp
  - p3.cpp
- Deberá subir sus archivos directamente a www.gradescope.com.
- Recuerde que Gradescope solo conserva el último envío que se realiza. Si va a modificar su entrega, debe volver a adjuntar todos los archivos de su práctica.

Competencias:

- Para los alumnos de la carrera de Ciencia de la Computación
  - Aplicar conocimientos de computación apropiados para la solución de problemas definidos y sus requerimientos en la disciplina del programa. (Evaluar)
  - Analizar problemas e identificar los requerimientos computacionales apropiados para su solución. (Usar)
  - Usar 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 necesidades deseadas dentro de restricciones realistas (nivel 2)

Calificación:

Tabla de puntos (Sólo para uso del profesor)

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

Preguntas

1. (7 puntos) Estructuras Selectivas:

Escriba un programa en C++ para clasificar instrumentos musicales antiguos basándose en diversas características. El programa deberá:

Solicitar al usuario ingresar detalles del instrumento, incluyendo:

- Periodo histórico (1: Antigüedad, 2: Edad Media, 3: Renacimiento)
- Material de fabricación (1: Madera, 2: Metal, 3: Cuero, 4: Combinado)
- Tipo de instrumento (1: Cuerda, 2: Viento, 3: Percusión)
- Región geográfica de origen (1: Asia, 2: Europa, 3: África, 4: Américas)
- Tamaño del instrumento (1: Pequeño, 2: Mediano, 3: Grande)

Clasificar el instrumento en categorías predefinidas basándose en las características ingresadas. Las categorías podrían incluir condiciones complejas y combinadas. Por ejemplo:

- Categoría I - "Liras Legendarias": Instrumentos de cuerda de madera, de la Antigüedad, originarios de Asia, tamaño pequeño o mediano.

- Categoría II - "Tambores Tribales": Instrumentos de percusión de cuero o combinados, de cualquier periodo histórico, originarios de África, tamaño grande.

- Categoría III - "Flautas de la Renovación": Instrumentos de viento de madera o metal, del Renacimiento, originarios de Europa o Américas, tamaño pequeño.

- Categoría IV - "Conjuntos del Cónclave": Instrumentos de cualquier tipo, material combinado, de la Edad Media, originarios de cualquier región, tamaño grande.

Para instrumentos que no encajen claramente en ninguna categoría, el programa debe identificarlos como "Categoría V - Instrumentos Atípicos" y proporcionar una salida indicando que el instrumento tiene características únicas que no cumplen con las categorías convencionales.

El programa deberá mostrar la categoría del instrumento junto con una breve descripción de las características lo conforman (excepto para la clasificación no sea posible debido a la combinación inusual de características, en ese caso no se describirán las características que lo conforman).

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado. El 100\% corresponde al puntaje indicado en cada punto.

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (100%).	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (70%).	El diseño tiene algunas deficiencias pero la ejecución es correcta (30%).	El diseño es deficiente y la ejecución no es correcta (0%).
Sintaxis	No existen errores sintácticos o de compilación (100%).	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (50%).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (30%).	El código tiene errores de sintáxis que afectan el resultado (10%).
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (100%).	El código es de buen performance durante la ejecución (70%).	El código no está optimizado pero la ejecución no es deficiente (30%).	El código no está optimizado y la ejecución es deficiente (0%).

2. (6 puntos) Funciones:

Desarrolle un programa en C++ que simule un sistema de análisis climático utilizando funciones. No puede usar arreglos ni vectores en esta pregunta.

El programa debe:

- Permitir al usuario ingresar una cantidad variable de datos climáticos diarios durante un período específico de días (desde el día 1 hasta el día ingresado por el usuario). Los datos a ingresar serán temperatura, humedad y presión atmosférica.

- Calcular y mostrar la media de temperatura, humedad y presión atmosférica.

- Identificar y mostrar los días con temperaturas, humedad o presión máxima y mínima (es decir, el número del día con máxima temperatura, el número de día con mínima temperatura, el número del día con máxima humedad, etc.).

- Finamente, presentar un resumen climático que incluya las medias, mínimos y máximos de cada dato climático.

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado. El 100\% corresponde al puntaje indicado en cada punto.

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (100%).	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (70%).	El diseño tiene algunas deficiencias pero la ejecución es correcta (30%).	El diseño es deficiente y la ejecución no es correcta (0%).
Sintaxis	No existen errores sintácticos o de compilación (100%).	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (50%).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (30%).	El código tiene errores de sintáxis que afectan el resultado (10%).
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (100%).	El código es de buen performance durante la ejecución (70%).	El código no está optimizado pero la ejecución no es deficiente (30%).	El código no está optimizado y la ejecución es deficiente (0%).

3. (7 puntos) punteros y arreglos:

Desarrolle un programa en C++ para gestionar eventos astronómicos. No puede usar vectores. El programa debe incluir las siguientes funcionalidades:

Lectura de cantidad de elementos a registrar:

Deberá usar el concepto de puntero, para apuntar a un arreglo dinámico del tamaño elegido por el usuario. Dicho tamaño debe ser ingresado por teclado al comenzar el programa.

Registro de Eventos:

Permita ingresar detalles como tipo de evento (1. eclipses, 2. lluvias de meteoritos, 3. tránsitos planetarios), fecha (dd/mm/aaaa), hora (hh:mm) y ubicación (coordenadas geográficas en pares ordenados para latitud y longitud).
Valide las fechas y horas ingresadas para garantizar que estén en formatos correctos.

Ingreso de Condiciones Climáticas y Predicción de Visibilidad:

Para cada evento registrado, ingresar las condiciones climáticas esperadas (1. claro, 2. parcialmente nublado, 3. nublado) y el nivel de iluminación ambiental y el nivel de iluminación ambiental (1. baja, 2. media, 3. alta).

Basado en estas condiciones, determine la visibilidad del evento (buena, moderada, pobre) según se indica en la siguiente tabla:

		Iluminación Ambiental		
		Baja	Media	Alta
Condiciones Climáticas	Claro	Buena	Buena	Moderada
	Parcialmente Nublado	Buena	Moderada	Pobre
	Nublado	Moderada	Pobre	Pobre

Impresión de reporte:

Implemente una función que permita imprimir un reporte con los eventos con Buena, Moderada o Pobre visibilidad, indicando su tipo de evento, fecha y ubicación. La visibilidad (Buena, Moderada o Pobre) debe ser un parámetro de esta función.

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado. El 100\% corresponde al puntaje indicado en cada punto.

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (100%).	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (70%).	El diseño tiene algunas deficiencias pero la ejecución es correcta (30%).	El diseño es deficiente y la ejecución no es correcta (0%).
Sintaxis	No existen errores sintácticos o de compilación (100%).	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (50%).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (30%).	El código tiene errores de sintáxis que afectan el resultado (10%).
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (100%).	El código es de buen performance durante la ejecución (70%).	El código no está optimizado pero la ejecución no es deficiente (30%).	El código no está optimizado y la ejecución es deficiente (0%).