UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II

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

Indicaciones específicas:

- Esta evaluación contiene 10 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:**

   Desarrolle un programa en C++ para clasificar especies de árboles en categorías únicas basadas en características específicas. El programa pedirá al usuario ingresar detalles sobre el árbol y, utilizando estructuras selectivas avanzadas, lo clasificará en una de las categorías especiales o determinará que es incategorizable.

   El usuario debe proporcionar la siguiente información:

   Altura del árbol: ¿El árbol es bajo (menos de 5 metros), mediano (entre 5 y 15 metros) o alto (más de 15 metros)? (Ingrese 1 para bajo, 2 para mediano, 3 para alto).

   Tipo de hoja: ¿Las hojas son caducas o perennes? (Ingrese 1 para caducas, 2 para perennes).

   Forma de la hoja: ¿Las hojas son simples, compuestas o aciculares (como agujas)? (Ingrese 1 para simples, 2 para compuestas, 3 para aciculares).

   Presencia de frutos: ¿El árbol tiene frutos? (Ingrese 1 para sí, 2 para no).

   Color de la flor: ¿El árbol tiene flores de color blanco, rojo, amarillo o no tiene flores? (Ingrese 1 para blanco, 2 para rojo, 3 para amarillo, 4 para ninguna).

   Clasificación de categorías:

   - **Categoría I - "Altos Perennes Fructíferos":** Árboles de más de 15 metros, con hojas perennes y frutos visibles, con flores blancas o sin flores.

   - **Categoría II - "Medianos Caducifolios Floridos":** Árboles entre 5 y 15 metros, con hojas simples o compuestas, sin frutos, con flores de cualquier color excepto rojo.

- **Categoría III - "Enanos Florales Efímeros":** Árboles de menos de 5 metros, con hojas caducas, con flores amarillas o sin flores.

- **Categoría IV - "Agujereados Policromáticos":** Árboles de cualquier altura con hojas aciculares y flores rojas o amarillas, o sin flores.

Si las características ingresadas no coinciden con ninguna de estas categorías o son contradictorias, el programa debe indicar: "Este árbol no se puede clasificar con la información proporcionada."

Algunos ejemplos de diálogo de este programa serían:

Ejemplo de ejecución 1:

```
Ingrese la altura del árbol (1: bajo, 2: mediano, 3: alto):
3

Ingrese el tipo de hoja (1: caducas, 2: perennes):
2

Ingrese la forma de la hoja (1: simples, 2: compuestas, 3:
aciculares):
1

Ingrese la presencia de frutos (1: sí, 2: no):
2

Ingrese el color de la flor (1: blanco, 2: rojo, 3: amari-
llo, 4: ninguna):
1

Este árbol se clasifica en la Categoría I - 'Altos Perennes
Fructíferos'
```

Ejemplo de ejecución 2:

```
Ingrese la altura del árbol (1: bajo, 2: mediano, 3: alto):
2

Ingrese el tipo de hoja (1: caducas, 2: perennes):
1
```

```
Ingrese la forma de la hoja (1: simples, 2: compuestas, 3:
aciculares):
2

Ingrese la presencia de frutos (1: sí, 2: no):
1

Ingrese el color de la flor (1: blanco, 2: rojo, 3: amari-
llo, 4: ninguna):
4

Este árbol no se puede clasificar con la información propor-
cionada.
```

Ejemplo de ejecución 3:

```
Ingrese la altura del árbol (1: bajo, 2: mediano, 3: alto):
1

Ingrese el tipo de hoja (1: caducas, 2: perennes):
2

Ingrese la forma de la hoja (1: simples, 2: compuestas, 3:
aciculares):
3

Ingrese la presencia de frutos (1: sí, 2: no):
2

Ingrese el color de la flor (1: blanco, 2: rojo, 3: amari-
llo, 4: ninguna):
2

Este árbol se clasifica en la Categoría IV - 'Agujereados
Policromáticos'.
```

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:

Escriba un programa en C++ para analizar las tendencias meteorológicas utilizando variables individuales para cada medición y una función de validación general. El programa debe incluir las siguientes funciones:

- **Función de validación:** Esta función recibe un valor, un mínimo y un máximo, y verifica si el valor está dentro del rango. Retorna 1 si el valor es válido y 0 si no lo es. Esta función se utilizará para validar las mediciones de temperatura, humedad y presión atmosférica.

- **Función para ingresar datos meteorológicos:** Permite al usuario ingresar la última, la penúltima y la antepenúltima medición para un tipo de dato meteorológico (temperatura, humedad, presión atmosférica), llamando a la función de validación para cada entrada.

- **Función para calcular la tendencia de cambio:** Determina si la tendencia de las tres mediciones ingresadas es creciente, decreciente o estable. Esta función analizará las tres mediciones y determinará la tendencia de la siguiente manera: Si cada medición es mayor que la anterior, la tendencia es creciente. Si cada medición es menor que la anterior, la tendencia es decreciente. Si las mediciones son iguales o no siguen un patrón claro, la tendencia es estable.

- **Función para generar un informe de tendencias:** Resume las tendencias de temperatura, humedad y presión atmosférica en un informe textual.

Algunos ejemplos de diálogo de este programa serían:

Ejemplo de ejecución 1:

```
Ingrese la última medición de temperatura: 20
Ingrese la penúltima medición de temperatura: 22
Ingrese la antepenúltima medición de temperatura: 24
Tendencia de temperatura: Decreciente

Ingrese la última medición de humedad: 45
Ingrese la penúltima medición de humedad: 44
Ingrese la antepenúltima medición de humedad: 46
Tendencia de humedad: Estable

Ingrese la última medición de presión atmosférica: 1012
Ingrese la penúltima medición de presión atmosférica: 1007
Ingrese la antepenúltima medición de presión atmosférica:
1003
Tendencia de presión atmosférica: Creciente

Informe de Tendencias:
Temperatura: Decreciente
Humedad: Estable
Presión atmosférica: Creciente
```

Ejemplo de ejecución 2:

```
Ingrese la última medición de temperatura: 16
Ingrese la penúltima medición de temperatura: 16
Ingrese la antepenúltima medición de temperatura: 15
Tendencia de temperatura: Estable

Ingrese la última medición de humedad: 70
Ingrese la penúltima medición de humedad: 65
Ingrese la antepenúltima medición de humedad: 60
Tendencia de humedad: Creciente

Ingrese la última medición de presión atmosférica: 995
Ingrese la penúltima medición de presión atmosférica: 1000
Ingrese la antepenúltima medición de presión atmosférica:
1005
Tendencia de presión atmosférica: Decreciente

Informe de Tendencias:
Temperatura: Estable
Humedad: Creciente
Presión atmosférica: Decreciente
```

Ejemplo de ejecución 3:

```
Ingrese la última medición de temperatura: 30
Ingrese la penúltima medición de temperatura: 28
```

```
Ingrese la antepenúltima medición de temperatura: 26
Tendencia de temperatura: Creciente

Ingrese la última medición de humedad: 40
Ingrese la penúltima medición de humedad: 42
Ingrese la antepenúltima medición de humedad: 44
Tendencia de humedad: Decreciente

Ingrese la última medición de presión atmosférica: 1010
Ingrese la penúltima medición de presión atmosférica: 1010
Ingrese la antepenúltima medición de presión atmosférica:
1010
Tendencia de presión atmosférica: Estable

Informe de Tendencias:
Temperatura: Creciente
Humedad: Decreciente
Presión atmosférica: Estable
```

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 estáticos:

Escriba un programa en C++ que simule un sistema para registrar los tiempos de tres nadadores en una competencia de natación, en diferentes estilos: Mariposa y Espalda. Además, compare estos tiempos con sus mejores marcas personales anteriores. El programa debe incluir las siguientes características, utilizando arreglos estáticos y punteros:

- **Ingreso de nombres y tiempos:** El programa debe permitir al usuario ingresar los nombres de los tres nadadores y sus tiempos actuales en cada estilo (mariposa y espalda), así como su mejor tiempo anterior en ese estilo. Use arreglos estáticos para almacenar los nombres y los tiempos. Valide que los tiempos sean números positivos.
- **Búsqueda del mejor tiempo por estilo y comparación con el récord personal:** Utilice punteros para encontrar el mejor tiempo actual en cada estilo (mariposa y espalda), y determinar si algún nadador ha mejorado su récord personal.
- **Cálculo del tiempo promedio por estilo:** Calcule y muestre el tiempo promedio de todos los nadadores para cada estilo (mariposa y espalda).
- **Determinar nadadores por encima del promedio y nadadores por debajo del promedio**: Muestre cuáles nadadores han logrado tiempos por encima del promedio y cuáles han logrado tiempos por debajo del promedio. En caso no haya nadadores por debajo o por encima del promedio, deberá especificarlo con un mensaje de salida.

El programa seguirá un flujo secuencial, primero ingresando los nombres y tiempos de los nadadores, luego realizando el análisis y presentando los resultados.

Algunos ejemplos de diálogo de este programa serían:

Ejemplo de ejecución 1:

```
Ingrese el nombre del primer nadador: Ana
Ingrese el nombre del segundo nadador: Bruno
Ingrese el nombre del tercer nadador: Carlos

Tiempos para el estilo mariposa:
Ingrese el tiempo actual de Ana en mariposa: 22.5
Ingrese el mejor tiempo anterior de Ana en mariposa: 23.0
Ingrese el tiempo actual de Bruno en mariposa: 24.1
Ingrese el mejor tiempo anterior de Bruno en mariposa: 24.0
Ingrese el tiempo actual de Carlos en mariposa: 21.8
Ingrese el mejor tiempo anterior de Carlos en mariposa: 22.0

Tiempos para el estilo espalda:
Ingrese el tiempo actual de Ana en espalda: 28.7
Ingrese el mejor tiempo anterior de Ana en espalda: 29.0
Ingrese el tiempo actual de Bruno en espalda: 27.5
Ingrese el mejor tiempo anterior de Bruno en espalda: 27.2
Ingrese el tiempo actual de Carlos en espalda: 26.9
```

```
Ingrese el mejor tiempo anterior de Carlos en espalda: 26.5

Mejor tiempo en mariposa: 21.8 segundos (Carlos)
Mejor tiempo en espalda: 26.9 segundos (Carlos)

Promedio de tiempos en mariposa: 22.8 segundos
Promedio de tiempos en espalda: 27.7 segundos

Nadadores por encima del promedio en mariposa: Ana
Nadadores por debajo del promedio en mariposa: Bruno, Carlos
Nadadores por encima del promedio en espalda: Ana
Nadadores por debajo del promedio en espalda: Bruno, Carlos
```

Ejemplo de ejecución 2:

```
Ingrese el nombre del primer nadador: Diana
Ingrese el nombre del segundo nadador: Eduardo
Ingrese el nombre del tercer nadador: Fabiola

Tiempos para el estilo mariposa:
Ingrese el tiempo actual de Diana en mariposa: 23.2
Ingrese el mejor tiempo anterior de Diana en mariposa: 23.5
Ingrese el tiempo actual de Eduardo en mariposa: 23.0
Ingrese el mejor tiempo anterior de Eduardo en mariposa:
22.8
Ingrese el tiempo actual de Fabiola en mariposa: 22.5
Ingrese el mejor tiempo anterior de Fabiola en mariposa:
22.7

Tiempos para el estilo espalda:
Ingrese el tiempo actual de Diana en espalda: 26.4
Ingrese el mejor tiempo anterior de Diana en espalda: 26.0
Ingrese el tiempo actual de Eduardo en espalda: 25.9
Ingrese el mejor tiempo anterior de Eduardo en espalda: 26.2
Ingrese el tiempo actual de Fabiola en espalda: 25.8
Ingrese el mejor tiempo anterior de Fabiola en espalda: 26.1

Mejor tiempo en mariposa: 22.5 segundos (Fabiola)
Mejor tiempo en espalda: 25.8 segundos (Fabiola)

Promedio de tiempos en mariposa: 22.9 segundos
Promedio de tiempos en espalda: 26.0 segundos

Nadadores por encima del promedio en mariposa: Eduardo,
Diana
Nadadores por debajo del promedio en mariposa: Fabiola
Nadadores por encima del promedio en espalda: Ninguno
```

```
Nadadores por debajo del promedio en espalda: Diana,
Eduardo, Fabiola
```

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%).