Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias

CONSIDERACIONES GENERALES

- El parcial es completamente individual
- El fraude puede ocasionar la apertura de un proceso disciplinario.
- Examen individual, sin materiales salvo lápiz, borrador y lapicero.
- Tiempo: 90 minutos. Puntaje total: 50 pts
- En preguntas de código: si se pide implementar funciones, **declare firma en operaciones.h** y escriba la implementación en el área solicitada (simule .cpp). Asuma que entradas y salidas estarán en main.cpp salvo que el enunciado pida otra cosa

"El estudiante de la Pontificia Universidad Javeriana, como agente de su propia formación, es corresponsable de la Identidad Institucional, uno de cuyos cimientos es tener como hábito un comportamiento ético en todos los ámbitos de la vida. En este sentido me comprometo a realizar con total integridad esta evaluación, solamente empleando los recursos autorizados para su desarrollo".
Consejo Académico, Acta Nro 79, abril 19 de 2004

Nombre: _____ Código: _____

Parte1 - Git y Github (5pts)

Ordene los pasos mínimos para editar main.cpp, registrar localmente y subir a GitHub

1. `git commit  "corrige condición de corte" -m`
2. `Editar main.cpp y guardar el archivo`
3. `Mover main.cpp a la raíz del disco`
4. `git push origin main`
5. `git pull origin main`
6. `git add main.cpp`
7. `git commit -m "corrige condición de corte`
8. `git origin pull main`

ORDEN: _____

Parte 2 – Lógica de programación básica con C++ (8 puntos)

La siguiente sección muestra cuatro códigos de programación que debería funcionar para lo siguiente

Leer calificaciones enteras (0–100) hasta que el usuario ingrese **-1**. Calcular el **promedio** de las calificaciones válidas (excluya el -1). Si no se ingresa ninguna calificación válida, el promedio debe ser **0**.

Indique para cada fragmento si es correcto o incorrecto y explique por qué.

A

```
int suma = 0;
int cantidad = 0;
int dato = 0;

while (dato != -1)
{
    cin >> dato;
    if (dato != -1)
    {
        suma = suma + dato;
        cantidad = cantidad + 1;
    }
}

double promedio = 0.0;
if (cantidad > 0)
{
    promedio = (1.0 * suma) / cantidad;
}
```

B

```
int suma = 0;
int cantidad = 0;
int dato = 0;

do
{
    cin >> dato;
    if (dato != -1)
    {
        suma = suma + dato;
        cantidad = cantidad + 1;
    }
}
while (dato != -1);

double promedio = 0.0;
if (cantidad > 0)
{
    promedio = (1.0 * suma) / cantidad;
}
```

C

```
int suma = 0;
int cantidad = 0;
int dato = 0;

cin >> dato;
while (dato != -1)
{
    suma = suma + dato;
    cantidad = cantidad + 1;
    cin >> dato;
}

double promedio = 0.0;
if (cantidad > 0)
{
    promedio = (1.0 * suma) / cantidad;
}
```

D

```
int suma = 0;
int cantidad = 0;
int dato = -1;

while (true)
{
    cin >> dato;
    if (dato == -1)
    {
        break;
    }
    suma = suma + dato;

}

double promedio = 0.0;
if (cantidad > 0)
{
    promedio = (1.0 * suma) / cantidad;
}
```

Código	Correcto porque	Incorrecto porque

Parte 3 Ejercicio de integración (37 pts)

Una célula del laboratorio necesita un programa de consola para registrar lecturas enteras de un sensor. La memoria disponible solo permite almacenar hasta 10 valores. El programa debe ofrecer un menú basado en switch para que el usuario pueda:

- 1. *Insertar* un entero al final del arreglo si hay espacio.
- 2. *Listar* en pantalla los valores almacenados desde el último elemento hacia el primero.
- 3. *Totales* →
    - `sumatoriaParcial`: calcular y mostrar la sumatoria de los elementos del arreglo que son mayores de 10 pero menores a 50,
    - `mayor:` calcular el mayor valor del arreglo. Si no se ha registrado ninguna lectura el valor del mayor es cero.
    - `total:` sumar la `sumatoriaParcial y el mayor.`

    Ejemplo:

    ```
    si arreglo = [5, 12, 60, 20]
    sumatoriaParcial  = 12 + 20 = 32,
    mayor = 60
    Total =  32 + 60
    ```
- 4. *Reiniciar* → vaciar el arreglo.
- 0. *Salir* del programa.

Restricciones y lineamientos obligatorios

1. Use **solo arreglos estáticos**: `int datos[10];` y la variable `int usados = 0;` para llevar el número de posiciones ocupadas.
2. Estructura en **tres archivos**: operaciones.h (declaraciones), operaciones.cpp (implementaciones) y main.cpp (orquestación y menú). CMakeLists.txt para compilar
3. Reglas de operación:
    - En la opción 1. si no hay espacio imprima `Sin espacio` y no modifique el arreglo. Si hay espacio, lea el valor y colóquelo en el arreglo de usados. Se permiten negativos y duplicados.
    - En la opción 2, si no hay datos imprima Vacio. En otro caso, imprima los datos de la posición final a la posición inicial.

- En la opción 4, ponga el valor de usados en cero.
- En la opción 5, termine el programa sin errores.

4. Validación de entrada: las opciones de menú se leen como números. Para cualquier otra opción, imprima `Opcion invalida` y muestre de nuevo el menú.

3ª (8 pts) Complete los huecos [A]..[D] para que la opción 1 (Insertar).

Inserte el entero al final, imprima Sin espacio si no hay lugar y actualice la variable usados que es el contador de cuántos espacios se han utilizado.

```
if ([A])
{
    int valor;
    cin >> valor;
    [B] = [D];
    [C];
}
else
{
    std::cout << "Sin espacio" << std::endl;
}
```

i.	usados < 10	vii.	valor
ii.	datos[usados]	viii.	usados = usados + 1
iii.	datos[0]	ix.	usados
iv.	datos[usados-1]	x.	usados = usados – 1
v.	usados <= 10	xi.	valor + 1
vi.	usados < 9	xii.	valor - 1

[A] _____ **[B]** _____ **[C]**_____ **[D]**_____

3b. (6pts) Complete los huecos [A]..[C] para implementar la opción 2 (Listar).

Listar en pantalla los valores almacenados desde el último elemento hacia el primero

```
for (int i = [A]; [B]; [C])
{
    std::cout << datos[i] << " ";
}
```

a. i = 0;
b. i = 1;
c. i = 10
d. i = 9;
e. i > 10
f. i >= 0;
g. i > 0;
h. i < 10;
i. i++;
j. i--;
k. i+=2;
l. i+=3;

[A] _____ **[B]** _____ **[C]**_____ **[D]**_____

3c. (20 pts) Implemente aquí las funciones para dar respuesta a la opción de **totales** declaradas en operaciones.h

Puntuación

```
#ifndef OPERACIONES_H
#define OPERACIONES_H

int calcularSumatoriaParcial(int datos[], int usados);
int calcularMayor(int datos[], int usados);
int calcularTotal(int datos[], int usados);

#endif
```

3d (3 pts) En el repositorio hay estos archivos en la raíz:

- main.cpp — contiene main() y llama a funciones declaradas en operaciones.h e implementadas en operaciones.cpp.
- operaciones.cpp — implementa funciones.
- operaciones.h — cabecera con prototipos.

Durante el intento de compilar el programa aparecen algunos errores. Empareje cada mensaje (1–3) con la causa más probable (A–D). Use cada letra una sola vez.

Mensajes:

1. fatal error: operaciones.h: No such file or directory
2. undefined reference to calcularMayor(int*, int, int)'

Causas:
A) El `add_executable` en CMake no incluye `operaciones.cpp`
B) operaciones.h no existe en la ruta indicada o el #include tiene error de nombre/ruta.
C) En el add_executable en CMake se incluyó el archivo operaciones.h y operaciones .cpp
D) En el add_executeen CMake se incluyó solo el archivo operaciones.cpp y main.cpp
E) Ninguna de estas explicaciones es adecuada para los mensajes de error

Emparejamiento:

1 → Causa : _____
2 → Causa: _____

Parte 5 – Ejercicio de programación (5pts)
La declaración de renta es el proceso anual en el que una persona informa a la autoridad tributaria sus ingresos, costos y deducciones, para determinar el impuesto a cargo. Para expresar montos de manera uniforme, se usa la Unidad de Valor Tributario (UVT), que es una cifra en pesos colombianos (COP) fijada para cada año.

Requerimiento funcional. La empresa desea una herramienta de consola que, dado un ingreso gravable en COP, calcule el impuesto a cargo:

- Valor **UVT**: **48,000 COP**.
- Calcule uvt como uvt = ingresoCop / UVT_VALOR usando **división entera**.
- Determine el impuesto **en UVT** con la siguiente tabla:

- 0–90 UVT: 0
- 91–180 UVT: 15% sobre el **exceso** de 90 UVT
- 181–400 UVT: **13.5 UVT** + 25% sobre el **exceso** de 180 UVT
- 400 UVT: **68.5 UVT** + 33% sobre el **exceso** de 400 UVT

- Convierta el resultado final a **COP** multiplicando por UVT_VALOR en el lugar adecuado y lo retorne.

Usted recibió los siguientes archivos
main.cpp
impuesto.cpp
impuesto.h

```
#include <iostream>
#include "impuesto.h"
using std::cin;
using std::cout;


int main()
{
	const int UVT_VALOR = 48000;
	long ingresoCop = 0;

	cin >> ingresoCop;

	long uvt = ingresoCop / UVT_VALOR;
	float impuestoUVT = calcularImpuestoUVT(uvt);

	float impuestoEnPesos = impuestoUVT * UVT_VALOR;
	long impuestoCop = (long) impuestoEnPesos;
	cout << “El impuesto a cargo es” << impuestoCop << endl;

	return 0;
}
```

Desarrolle la implementación de la función

```
	float calcularImpuestoUV(uvt) (5 pts Bonus)
```