UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 1
Pregrado
2022-I
Profesor Jorge Villavicencio
Lab 1.07

Indicaciones específicas:

- Esta evaluación contiene 12 páginas (incluyendo esta página) con 3 preguntas. El total de puntos son 20.
- El tiempo límite para la evaluación es 100 minutos.
- Cada pregunta deberá ser respondida en un solo archivo con el número de la pregunta.
  - `p1.cpp`
  - `p2.cpp`
  - `p3.cpp`
- No se permiten tildes en los resultados en consola.
- Recuerda que el Gradescope solo conserva el último envio que se realiza, por lo tanto una vez que tengas las 3 preguntas resueltas, **deberás arrastrar los 3 archivos de manera simultánea y subirlos al Gradescope**.
  www.gradescope.com

Competencias:

- Para los alumnos de la carrera de Ciencia de la Computación

  Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (Evaluar)

  Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución.(Usar)

  Utilizar 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 las necesidades deseadas dentro de restricciones realistas (nivel 2)

Calificación:

Tabla de puntos (sólo para uso del professor)

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

1. (7 points) **Mayor ataque y promedio Pokemón**
Se tiene la siguiente tabla de pokemones:

id	Nombre Pokemón
1	Pikachu
2	Charmander
3	Bulbasaur
4	Squirtle
5	Butterfree
6	Arbok
7	Jigglypuff
8	Meowth
9	Psyduck
10	Rapidash

Se requiere crear un programa que permita el ingreso de un número "n", el cual representa la cantidad de veces que se solicitará el id, valor de ataque y valor de defensa por pokemon.

Luego con los datos ingresados se solicita hallar el promedio de los ataques y defensas. Adicionalmente indicar el nombre del pokemón que tiene mayor ataque.

Consideraciones:

- El valor de la variable "n"tiene que ser mayor a 1 y menor igual a 10, en caso el usuario ingrese un valor fuera del rango, el programa debe solicitar ingresar nuevamente el valor.
- En caso el id del pokemon con mayor ataque no se encuentre en el rango de 1 a 10 deberá mostrarse el siguiente mensaje en consola: 'Pokemon no registrado por la Pokedex"

Listing 1: Ejemplo 1

```
Cantidad de pokemones [ 2 - 10] = 4
Numero de pokemon = 4
Ataque = 100
Defensa = 50
Numero de pokemon = 2
Ataque = 20
Defensa = 80
Numero de pokemon = 3
Ataque = 28
Defensa = 32
Numero de pokemon = 9
Ataque = 10
Defensa = 10
```

```
Promedio ataque = 39.5
Promedio defensa = 43
Pokemon con mayor ataque es: Squirtle
```

Listing 2: Ejemplo 2

```
Cantidad de pokemones [ 2 - 10] = 0
Cantidad de pokemones [ 2 - 10] = -3
Cantidad de pokemones [ 2 - 10] = 13
Cantidad de pokemones [ 2 - 10] = 11
Cantidad de pokemones [ 2 - 10] = 2
Numero de pokemon = 8
Ataque = 29
Defensa = 10
Numero de pokemon = 1
Ataque = 90
Defensa = 20
Promedio ataque = 59.5
Promedio defensa = 15
Pokemon con mayor ataque es: Pikachu
```

Listing 3: Ejemplo 3

```
Cantidad de pokemones [ 2 - 10] = 0
Cantidad de pokemones [ 2 - 10] = 2
Numero de pokemon = 10
Ataque = 30
Defensa = 40
Numero de pokemon = 17
Ataque = 90
Defensa = 10
Promedio ataque = 60
Promedio defensa = 25
Pokemon con mayor ataque es: Pokemon no registrado por la
   Pokedex
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (`3pts`)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (`2pts`)	La ejecución es correcta (`1pts`).	La ejecución no es correcta (`0.5pts`)
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (`1.5pts`).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (`1pts`).	El código tiene errores de sintáxis que afectan el resultado (`0.5pts`).
Optimizacion	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`2pts`)	El codigo es de buen performance durante la ejecución (`1.5pts`).	El codigo no está optimizado pero la ejecución no es deficiente(`1pts`).	El codigo no está optimizado y la ejecución es deficiente (`0pts`).

2. (6 points) **Sumatoria de cubos recursivo.**
Tenemos la sumatoria de cubos representado en la siguiente formula

$$\sum_{i=1}^{n} x_i^3 = 1^3 + 2^3 + 3^3 + 4^3 + ... + n^3$$

Se solicita crear un programa que calcule la sumatoria de los números elevados al cubo desde 1 hasta"n", por ejemplo si n = 6 el resultado es 441. **Crear una función recursiva que permita calcular la sumatoria de cubos.**

Consideraciones:

- El rango de valores que puede ingresar para la variable n es entre 0 y 80. El programa debe validar el valor del dato, en caso sea un número fuera del rango se debe volver a solicitar su ingreso

Listing 4: Ejemplo 1

```
n = 6
Cubos(6) = 441
```

Listing 5: Ejemplo 2

```
n = 1
Cubos(1) = 1
```

Listing 6: Ejemplo 3

```
n = 80
Cubos(80) = 10497600
```

Listing 7: Ejemplo 4

```
n = 40
F(40) = 102334155
```

Listing 8: Ejemplo 5

```
n = -5
n = -9
n = 90
n = 145
n = 7
Cubos(7) = 784
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (`2pts`)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (`1.5pts`)	La ejecución es correcta (`1pts`).	La ejecución no es correcta (`0.5pts`)
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (`1.5pts`).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (`1pts`).	El código tiene errores de sintáxis que afectan el resultado (`0.5pts`).
Optimizacion	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`2pts`)	El codigo es de buen performance durante la ejecución (`1.5pts`).	El codigo no está optimizado pero la ejecución no es deficiente(`1pts`).	El codigo no está optimizado y la ejecución es deficiente (`0pts`).

3. (7 points) **Calificación de estrellas para películas de Marvel.**
Te han pedido realizar un programa que pueda calificar 4 peliculas del universo de Marvel (Doctos Strange, Capitán América, Wanda y Loki) para lo cual debes **implementar los prototipos de funciones usando punteros**, los cuales se myuestran a continuación:

Listing 9: Prototipo de Funciones

```
void actualizaDoctorStrange(int *ptosDoctor);
void actualizaCapitan(int *ptosCapitan);
void actualizaWanda(int *ptosWanda);
void actualizaLoki(int *ptosLoki);
void imprimirPrimerPuesto(int *ptosDoctor, int *ptosCapitan,
    int *ptosWanda,int *ptosLoki);
void menu(int &opt);
```

Consideraciones:

- Está permitido crear más funciones si el alumno considera que el ejercicio lo requiere, pero como mínimo debe usar las funciones prototipo propuestas.
- Las funciones que tienen el prefijo "actualiza", es una función que tiene como parámetro por referencia un puntero, se debe hacer uso de punteros para actualizar la cantidad de estrellas por cada película.
- "imprimirPrimerPuesto", debe evaluar los valores de los punteros para imprimir cual es la película con mayor puntuación de estrellas.
- Se brinda parte del código de la función Menu, se solicita modificar para que el usuario solo ingrese valores del 1 al 6 en caso que ingrese números fuera del rango, se debe volver a mostrar el menú.

Listing 10: Función Menú

```
void menu(int &opt){
        cout<<" MENU"<<"\n";
        cout<<" ----"<<"\n";
        cout<<"1.Calificar Doctor Strange"<<"\n";
        cout<<"2.Calificar Capitan America"<<"\n";
        cout<<"3.Calificar WandaVision"<<"\n";
        cout<<"4.Calificar Loki"<<"\n";
        cout<<"5.Imprimir primer puesto"<<"\n";
        cout<<"6.SALIR"<<"\n\n";
        cout<<"Ingresar opcion (1-6): ";
        cin>>opt;
}
```

- Se debe mostrar el menú hasta que el usuario no presione la opción 6.SALIR, con esta opción se da por terminado el programa. Considerar esta validación en el main.

Listing 11: Ejemplo 1

```
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 1
Ingrese puntos para Doctor Strange: 20
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 2
Ingrese puntos para Capitan America: 30
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 4
Ingrese puntos para Loki: 12
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 3
```

```
Ingrese puntos para Wanda: 90
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 5
 TABLA DE PUNTAJES
 -----------------
Primer puesto: Wanda con 90
 -----------------
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 6
```

Listing 12: Ejemplo 2

```
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 8
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
```

```
6.SALIR

Ingresar opcion (1-6): 10
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 1
Ingrese puntos para Doctor Strange: 40
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 4
Ingrese puntos para Loki: 10
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 5
 TABLA DE PUNTAJES
 -----------------
Primer puesto: Doctor Strange con 40
 -----------------
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
```

```
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 6
```

Listing 13: Ejemplo 3

```
 MENU
 ----
1.Calificar Doctor Strange
2.Calificar Capitan America
3.Calificar WandaVision
4.Calificar Loki
5.Imprimir primer puesto
6.SALIR

Ingresar opcion (1-6): 6
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (3pts)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (2pts)	La ejecución es correcta (1pts).	La ejecución no es correcta (0.5pts)
Sintaxis	No existen errores sintácticos o de compilación (2pts)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (1.5pts).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (1pts).	El código tiene errores de sintáxis que afectan el resultado (0.5pts).
Optimizacion	El código es óptimo y eficiente. De buen performance e interacción con el usuario (2pts)	El codigo es de buen performance durante la ejecución (1.5pts).	El codigo no está optimizado pero la ejecución no es deficiente(1pts).	El codigo no está optimizado y la ejecución es deficiente (0pts).