UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 1
Pregrado
2022-II
Profesor: Ian Paul Brossard
Lab: 2.07

Indicaciones específicas:

- Esta evaluación contiene 8 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`
- Deberás subir estos archivos directamente a www.gradescope.com, uno en cada ejercicio. También puedes crear un .zip

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) **Uno, uno, cuadrado:**
Elaborar un programa en lenguaje C++ que imprima los n primeros términos de la sucesión 1, 1, 4, 1, 1, 25, 1, 1, 64, 1, 1...

Para elaborar esta sucesión, deberá utilizar una estructura iterativa. Los términos que se impriman en esta sucesión son 1, 1, y el cuadrado del total de elementos impresos hasta el momento.

Es decir, esta sucesión imprime 1, luego otro 1, y luego el cuadrado de 2 (debido a que hasta el momento solo se han impreso dos términos: 1, 1). Luego, se imprime 1, luego otro 1, y luego el cuadrado de 5 (debido a que hasta el momento se han impreso cinco términos: 1, 1, 4, 1 y 1). Luego, se imprime 1, luego otro 1, luego el cuadrado de 8 (debido a que hasta el momento se han impreso los ocho términos: 1, 1, 4, 1, 1, 25, 1 y 1).

Asuma que el usuario siempre ingresará una cantidad de términos positiva. Es decir, no es necesario validar que el número de términos no sea cero ni negativo.

Para calcular el cuadrado de un número, basta con multiplicarlo consigo mismo.

Ejemplo de ejecución 1:

```
Ingrese el número de términos: 10
1
1
4
1
1
25
1
1
64
1
```

Ejemplo de ejecución 2:

```
Ingrese el número de términos: 4
1
1
4
1
```

Ejemplo de ejecución 3:

```
Ingrese el número de términos: 1
1
```

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) **Sombrero y suela de zapato:**
Elaborar dos funciones: una función llamada sombrero y otra función llamada suelaDeZapato. Ambas funciones recibirán un valor double n y retornarán un entero, y funcionarán de la siguiente manera:

La función sombrero retornará el entero inmediato mayor o igual n, y la función suelaDeZapato retornará el entero inmediato menor o igual a n.

Desde el el programa principal *main* se deberá leer una variable de tipo double x, y utilizar las funciones sombrero y suelaDeZapato para imprimir al entero inmediato mayor o igual a x y al entero inmediato menor o igual a x.

Asuma que x siempre es positivo. Es decir, no debe validar que x sea positivo, ni tampoco debe considerar el valor sombrero y suela de zapato para números negativos.

Recuerde que puede obtener la parte entera de un valor real escribiendo (int) delante de una variable de tipo double. Por ejemplo, la sentencia a = (int)x; almacena la parte entera de x en la variable a. Por lo tanto, si x tuviera el valor de 2.7, entonces dicha sentencia almacenaría 2 en a.

Ejemplo de ejecución 1:

```
Ingrese x: 2.1
el entero inmediato mayor o igual a x es: 3
el entero inmediato menor o igual a x es: 2
```

Ejemplo de ejecución 2:

```
Ingrese x: 2
el entero inmediato mayor o igual a x es: 2
el entero inmediato menor o igual a x es: 2
```

Ejemplo de ejecución 3:

```
Ingrese x: 5.99
el entero inmediato mayor o igual a x es: 6
el entero inmediato menor o igual a x es: 5
```

Ejemplo de ejecución 4:

```
Ingrese x: 10.001
el entero inmediato mayor o igual a x es: 11
el entero inmediato menor o igual a x es: 10
```

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) **Buenos días señorita, buenos días caballero:**
Elaborar un programa en lenguaje C++ que lea un valor entero n (asuma que el valor de *n* siempre es positivo, es decir, no es necesario que controle los casos negativos o cero).

Una vez leído el valor de *n*, su programa principal deberá hacer un llamado a una función recursiva que le permita imprimir todos los números desde 1 hasta *n*, acompañados de un cordial saludo (ver ejemplos de ejecución).

Cada vez que imprima un número, las función recursiva deberá verificar si se trata de un número impar o par. Si se trata de un número impar, la función recursiva imprimirá el número, y el mensaje "buenos días señorita". En cambio, si se trata de un número par, su función recursiva imprimirá el número y el mensaje "buenos días caballero".

Una vez que se impriman los *n* números con su respectivo saludo, se deberá imprimir el mensaje "que empiece el baile", tal y como se aprecia en los ejemplos de ejecución.

Se recomienda que la primera vez que sea invocada esta función recursiva, imprima el número 1 y, dado que 1 es impar, imprima también el mensaje "buenos días señorita". Luego, evalúe si dicho número 1 impreso es igual a *n*. Si el número impreso no es igual a *n*, entonces la función recursiva se volverá a llamar a sí misma. En la segunda recursión, la función imprimirá el número 2 y, dado que 2 es par, imprimirá también el mensaje "buenos días caballero" y, nuevamente, evaluará si ese número 2 impreso es igual a n. Si no es igual a n, repetirá el proceso recursivamente hasta que en algún momento la función imprima un número igual a n.

Pista: dado que su función necesita conocer el valor de *n* y necesita saber qué número toca imprimir en cada recursión, seguramente esta función deberá trabajar con 2 parámetros.

Ejemplo de ejecución 1:

```
ingrese n: 8
1 buenos días as señorita
2 buenos días caballero
3 buenos días as señorita
4 buenos días caballero
5 buenos días as señorita
6 buenos días caballero
7 buenos días as señorita
8 buenos días caballero
¡Que empiece el baile!
```

Nota: En este ejercicio aparecen las palabras *señorita* y *caballero* debido a que se trata de versión simplificada de otro ejercicio de programación que simulaba el popular juego de cartas *Nervioso*.

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`).