UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación Orientada a Objetos 1

Práctica Calificada 1
Pregrado
2020-II
Profesor Rubén Rivas
Lab 206

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 y tu código de estudiante. Por ejemplo:
  1. `p1.h` y `p1.cpp`
  2. `p2.h` y `p2.cpp`
  3. `p3.h` y `p3.cpp`
- Deberás subir estos archivos directamente a www.gradescope.com, uno en cada ejercicio.

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	6	
2	7	
3	7	
Total:	20	

1. (6 points) **Cajero**
   Con el fin de dar facilidades a los clientes de un banco en el retiro del dinero y mínimizar el número de billetes, se propone que el importe de un retiro sea subdividido en billetes considerando primero los billetes de mayor denominación e ir subdividiendolo en billetes de menor denominación.

Billete
200 soles
100 soles
50 soles
20 soles
10 soles
5 soles
2 soles
1 soles

   Escribir un programa que solicite un monto de retiro y que retorne la cantidad de billetes por denominacion.

   Algunos ejemplos de diálogo de este programa serían:

Listing 1: Ejemplo 1

```
Ingrese el retiro: 1500
7 billete(s) de 200 soles
1 billete(s) de 100 soles
```

Listing 2: Ejemplo 1

```
Ingrese el retiro: 72
1 billete(s) de 50 soles
1 billete(s) de 20 soles
1 moneda(s) de 2 soles
```

Listing 3: Ejemplo 1

```
Ingrese el retiro: 388
1 billete(s) de 200 soles
1 billete(s) de 100 soles
1 billete(s) de 50 soles
1 billete(s) de 20 soles
1 billete(s) de 10 soles
1 moneda(s) de 5 soles
1 moneda(s) de 2 soles
1 moneda(s) de 1 soles
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Código	Ha implementado las estructuras de control necesarias para resolver el problema (`3pts`)	Existen algunos errores menores en la implementación (`2pts`)	El diseño y la implementacion del código no son correctos (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El codigo es optimizable en algunas partes (`0.5pts`).	El codigo es redundante y/o no es optimo (`0pts`).

2. (7 points) **Cuadrados Concentricos**
   Diseñar y escribir un programa que solicite un número entero n y que dibuje 2 rectángulos concentricos donde el rectángulo externo es de lado n y el rectángulo interno es de lado 2 si el valor de n sea par o de lado 1 si es impar.

   Algunos ejemplos de diálogo de este programa serían:

Listing 4: Ejemplo 1

```
Ingrese n: 5
* * * * *
*       *
*   +   *
*       *
* * * * *
```

Listing 5: Ejemplo 2

```
Ingrese n: 6
* * * * * *
*         *
*   + +   *
*   + +   *
*         *
* * * * * *
```

Listing 6: Ejemplo 3

```
Ingrese n: 9
* * * * * * * * *
*               *
*               *
*               *
*       +       *
*               *
*               *
*               *
* * * * * * * * *
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Código	Ha implementado las estructuras de control necesarias para resolver el problema (`3pts`)	Existen algunos errores menores en la implementación (`2pts`)	El diseño y la implementacion del código no son correctos (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`3pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El codigo es optimizable en algunas partes (`0.5pts`).	El codigo es redundante y/o no es optimo (`0pts`).

3. (7 points) **Conjetura de Collatz**
La conjetura de Collatz consiste en aplicar operaciones a un número entero positivo siguiendo estas dos condiciones:

   - Si el número es par se divide en 2.
   - Si el número es impar se multiplica por 3 y al resultado se le suma 1

   lo interesante es que si se ejecuta repetidamente estas 2 operaciones el resultado final es 1 y aunque son simples las reglas hasta ahora no hay quien pueda demostrar que se cumple para cualquier número entero positivo.

   Se solicita crear la función **contar_repeticiones_collatz** que reciba a través de un párametro un número entero positivo y por medio de otro párametro del tipo puntero retornar las veces en que se ejecuta las condiciones mencionadas al inicio de la pregunta hasta que el número llegue a 1.

   Algunos ejemplos de diálogo de este programa serían:

Listing 7: Ejemplo 1

```
Ingrese un numero: 20
# de Operaciones: 7
// 10, 5, 16, 8, 4, 2, 1
```

Listing 8: Ejemplo 1

```
Ingrese un numero: 9
# de Operaciones: 19
// 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5,
   16, 8, 4, 2, 1
```

Listing 9: Ejemplo 1

```
Ingrese un numero: 2
# de Operaciones: 1 // 1
```

Listing 10: Ejemplo 1

```
Ingrese un numero: 3
# de Operaciones: 7
// 10, 5, 16, 8, 4, 2, 1
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Código	Ha implementado funciones y punteros en forma correcta y lógica (`3pts`)	Existen algunos errores menores en la implementación (`2pts`)	El diseño y la implementacion del código no son correctos (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`3pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El codigo es optimizable en algunas partes (`0.5pts`).	El codigo es redundante y/o no es optimo (`0pts`).