UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2-deportista
Pregrado
2022-II
Profesor: Angel Napa
Lab 2.07

Indicaciones específicas:

- Esta evaluación contiene 11 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) **Matrices**
   Printeo
   Por la semana UTEC, el profesor intentó crear un programa que creara estampados cuadriculares de las siglas de la universidad para la PC del curso de programación 2 para el día sábado. El detalle es que decidió que el programa solo imprima uno de las letras de la sigla para que los estudiantes tengan tiempo de revisar los 3 problemas.

   Se solicita al alumno que cree un programa que realice lo siguiente: reciba un valor n del usuario. Debe solicitarlo hasta que el usuario ingrese un **entero positivo impar mayor a 5**.Luego de esto el programa debe solicitar al usuario de qué letra quiere crear un estampado. El programa deberá validar que el usuario ingrese uno de los siguientes 4 char: 'U', 'T', 'E' y 'C'. Con esta información el programa deberá imprimir una matriz de $n \times n$. donde el borde externo de la matriz esté "vacío"(esto significa '.'), y dentro debe aparecer la letra solicitada por el usuario. El estampado estará representado por el char '#'. El grosor de las letras son de 1 unidad, y el ancho y altura de la letra debe ser de $n - 2$.
   Para que se consiga el puntaje completo el programa deberá:

   - Utilizar punteros dinámicos para la creación de la matriz
   - Liberar la memoria del heap.

   El estudiante es libre de usar temas anteriores a matrices para la resolución de este problema(funciones, variables globales, referencia, iteraciones, etc.). No es necesario que los mensajes string del terminal sean exactamente como mostrado en la hoja.

```
Ingrese tamanho de la matriz: 4
Tamanho no valido.
Ingrese tamanho de la matriz: 5
Tamanho no valido.
Ingrese tamanho de la matriz: 6
Tamanho no valido.
Ingrese tamanho de la matriz: 7
ingrese letra a estampar(U, T, E o C):a
Letra no valida
ingrese letra a estampar(U, T, E o C):c
Letra no valida
ingrese letra a estampar(U, T, E o C):C
matriz generada:
. . . . . .
. # # # # # .
. # . . . . .
. # . . . . .
. # . . . . .
. # # # # # .
. . . . . .
```

```
Ingrese tamanho de la matriz: 7
ingrese letra a estampar(U, T, E o C):u
Letra no valida
ingrese letra a estampar(U, T, E o C):U
matriz generada:
. . . . . .
. # . . . # .
. # . . . # .
. # . . . # .
. # . . . # .
. # # # # # .
. . . . . .
```

```
Ingrese tamanho de la matriz: 11
ingrese letra a estampar(U, T, E o C):T
matriz generada:
. . . . . .
. # # # # # # # # # .
. . . . . # . . . . .
. . . . . # . . . . .
. . . . . # . . . . .
. . . . . # . . . . .
. . . . . # . . . . .
. . . . . # . . . . .
. . . . . # . . . . .
. . . . . # . . . . .
. . . . . .
```

```
Ingrese tamanho de la matriz: 9
ingrese letra a estampar(U, T, E o C):E
matriz generada:
. . . . . .
. # # # # # # # .
. # . . . . . .
. # . . . . . .
. # # # # # # # .
. # . . . . . .
. # . . . . . .
. # # # # # # # .
. . . . . .
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Algoritmo	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`)
Código	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`).
Eficiencia	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) **Vectores**
Suma de cuadrados
Problema de enunciado simple. Se pide construir un programa que reciba un entero positivo n (no es necesario validarlo: asumir que el usuario siempre ingresará un entero positivo), junto con n enteros positivos que varian entre 1 a 1000 (no es necesario validar), que se guardarán en un vector. El programa debe alterar el vector construido de modo que se eliminen todos los elementos de posición divisibles entre 5. Debe imprimir el vector resultante y calcular la suma de todos los cuadrados perfectos que se encuentren en este vector resultante.

```
Ingrese tamanho de vector:5
Ingrese los 5 elementos del vector:
1 2 3 4 5
vector de elementos en posicion multiplos de 5:
2 3 4 5
Suma de cuadrados perfectos de este vector:
4
```

```
Ingrese tamanho de vector:10
Ingrese los 10 elementos del vector:
2 3 5 7 11 13 17 19 23 29
vector de elementos en posicion multiplos de 5:
3 5 7 11 17 19 23 29
Suma de cuadrados perfectos de este vector:
0
```

```
Ingrese tamanho de vector:7
Ingrese los 7 elementos del vector:
1 4 9 16 25 36 49
vector sin los elementos de posicion multiplos de 5:
4 9 16 25 49
Suma de cuadrados perfectos de este vector:
103
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Algoritmo	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`)
Código	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`).
Eficiencia	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) **Clases**
Espacio tridimensional
Se pide construir una clase CPoint que tenga como atributos 3 numeros tipo float, junto con todas las funciones trabajadas en clase(setters, getters, constructores y destructor). Sumado a eso se debe construir funciones (ya sea dentro de la clase o aparte) que realicen lo siguiente:

   1. La funcion *suma* tendrá como parámetros 2 CPoint y devolverá un 1 CPoint que resulte de sumar los 2 parametros (tal cual como espacio tridimensional)
   2. La funcion *resta* tendrá como parámetros 2 CPoint y devolverá un 1 CPoint que resulte de restar los 2 parametros (tal cual como espacio tridimensional)
   3. La funcion *dotproduct* tendrá como parámetros 2 CPoint y devolverá un 1 double que resulte de realizar el producto escalar. Recordemos que el producto escalar de los puntos (a, b, c) y (d, e, f) es el escalar $(a, b, c) \cdot (d, e, f) = ad + be + cf$.
   4. La función *crossproduct* tendrá como parámetros 2 CPoint y devolverá 1 CPoint que resulte de realizar el producto vectorial. Recordemos que el producto vectorial de 2 vectores (a, b, c) y (d, e, f) es otro vector $(a, b, c) \times (d, e, f) = (bf - ce, cd - af, ae - bd)$

5. La funcion *distancia* que reciba como parámetro un CPoint y devuelva la distancia entre el punto y el origen $(0, 0)$. Recordemos que la distancia entre (a, b, c) y $(0, 0)$ es $d = \sqrt{(a, b, c) \cdot (a, b, c)} = \sqrt{a^2 + b^2 + c^2}$

Escriba un programa main que realice lo siguiente:

- Solicite al programa las coordenadas de 3 puntos en el plano cartesiano
- Guarde estos datos en 2 objetos de tipo CPoint: p1 y p2.
- Luego, el programa debe solicitar al usuario uno de los siguientes char: '+', '-', '.','x' y 'd', hasta que el usuario ingrese '0'.
- Si el usuario ingresa un caracter diferente a los 4 mencionados arriba, solicitará la lectura de un nuevo caracter.
- Cada vez que el usuario ingrese una de las operaciones válidas, el usuario devolverá el resultado de realizar dicha operación con los puntos p1 y p2: + devolverá la suma p1+p2; '-' devuelve la resta; '.', el producto escalar; 'x', el producto cruz; y 'd', la distancia entre los puntos p1 y p2.

Aclaración: son libres de ingresar más de un parámetro en las funciones mencionadas. Esto depende de si quisieran trabajar con variables globales, referencia, punteros, etc. Lo que sí es obligatorio es que ambas funciones llamen a un objeto tipo CPoint.

```
Ingrese coordenadas del primer punto: 0 0 0
Ingrese coordenadas del primer punto: 6 1.5 2
Las coordenadas de p1 son:
(0, 0, 0)
Las coordenadas de p2 son:
(6, 1.5, 2)
Ingrese operacion(+, -, ., x, d, 0 para cerrar): d
La distancia entre p1 y p2 es:
6.5
Ingrese operacion(+, -, ., x, d, 0 para cerrar): 0
Gracias
```

```
Ingrese coordenadas del primer punto: 1 0 0
Ingrese coordenadas del primer punto: 0 1 0
Las coordenadas de p1 son:
(1, 0, 0)
Las coordenadas de p2 son:
(0, 1, 0)
Ingrese operacion(+, -, ., x, d, 0 para cerrar): r
caracter invalido
Ingrese operacion(+, -, ., x, d, 0 para cerrar): +
La suma de p1 y p2 es:
(1, 1, 0)

Ingrese operacion(+, -, ., x, d, 0 para cerrar): -
La resta de p1 y p2 es:
(1, -1, 0)

Ingrese operacion(+, -, ., x, d, 0 para cerrar): .
El producto punto de p1 y p2 es:
0
Ingrese operacion(+, -, ., x, d, 0 para cerrar): x
El producto cruz de p1 y p2 es:
(0, 0, 1)
Ingrese operacion(+, -, ., x, d, 0 para cerrar): d
La distancia entre p1 y p2 es:
1.41421
Ingrese operacion(+, -, ., x, d, 0 para cerrar): 0
Gracias!
```

```
Ingrese coordenadas del primer punto: 0 1 0
Ingrese coor-denadas del primer punto: 1 0 0
Las coordenadas de p1 son:
(0, 1, 0)
Las coordenadas de p2 son:
(1, 0, 0)
Ingrese operacion(+, -, ., x, d, 0 para cerrar): x
El producto cruz de p1 y p2 es:
(0, 0, -1)
Ingrese operacion(+, -, ., x, d, 0 para cerrar):
0
Gracias
```

```
Ingrese coordenadas del primer punto: 1 2 3
Ingrese coordenadas del primer punto: 4 2.5 0
Las coordenadas de p1 son:
(1, 2, 3)
Las coordenadas de p2 son:
(4, 2.5, 0)
Ingrese operacion(+, -, ., x, d, 0 para cerrar): .
El producto punto de p1 y p2 es:
9
Ingrese operacion(+, -, ., x, d, 0 para cerrar): 0
Gracias
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Algoritmo	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`)
Código	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`).
Eficiencia	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`).