UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2
Pregrado
2022-II
Profesor: Rubén Rivas
Lab: 106

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, p1.h`
  - `p2.cpp, p2.h`
  - `p3.cpp, p3.h`
- 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 apropiados para la solución de problemas definidos y sus requerimientos en la disciplina del programa. (nivel 2)

  Diseñar, implementar y evaluar soluciones a problemas complejos de computación.(nivel 2)

  Crear, seleccionar, adaptar y aplicar técnicas, recursos y herramientas modernas para la práctica de la computación y comprende sus limitaciones. (nivel 2)

- Para los alumnos de las carreras de Ingeniería

  Aplicar conocimientos de ingeniería en la solución de problemas complejos de ingeniería (nivel 2).

  Diseñar soluciones relacionados a problemas complejos de ingeniería (nivel 2)

  Crear, seleccionar y utilizar técnicas, habilidades, recursos y herramientas modernas de la ingeniería y las tecnologías de la información, incluyendo la predicción y el modelamiento, con la comprensión de sus limitaciones (nivel 2)

- Para los alumnos de Administración y Negocios Digitales

  Analizar información verbal y/o lógica proveniente de distintas fuentes, encontrando relaciones y presentándola de manera clara y concisa (nivel 2)

Analizar y evaluar el comportamiento del consumidor y el desarrollo de estrategias comerciales (nivel 2)

Trabajar de manera efectiva con equipos multidisciplinarios y diversos en género, nacionalidad, edad, etc. (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) **Producto de matrices**
Escribir un programa que utilizando matrices dinámicas, lea desde el teclado dos matrices de números enteros de tamaño $N \times N$, y que genere una nueva matriz que sea el calculo del producto de las 2 matrices.

Ejemplo: Para $N = 3$ las 2 matrices seria de 3×3:

$$\begin{bmatrix} a_{11} & ... & a_{13} \\ ... & ... & ... \\ a_{31} & ... & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & ... & b_{13} \\ ... & ... & ... \\ b_{31} & ... & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & ... & c_{13} \\ ... & ... & ... \\ c_{31} & ... & c_{33} \end{bmatrix}$$

donde:
$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$
$c_{13} = a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33}$
$c_{31} = a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31}$
$c_{33} = a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33}$

Caso de uso #1 Input:

```
3          // Valor n
2 2 2      // Inicio de primera matriz
3 3 3
4 4 4
1 2 3      // Inicio de segunda matriz
1 2 3
1 2 3
```

Output:

```
 6   12   18
 9   18   27
12   24   36
```

Caso de uso #2 Input:

```
6
1 4 5 6 4 3
4 3 3 4 8 1
6 3 8 5 9 3
5 9 3 6 1 7
4 8 3 6 2 1
9 5 6 3 1 0
5 6 4 5 1 5
1 7 2 8 4 5
4 3 5 2 9 1
7 4 6 5 5 8
5 1 8 2 6 3
8 3 6 4 6 5
```

Output:

```
 115   86  123   97  134  105
 111   81  131   90  117   99
 169  119  190  125  187  135
 149  148  139  163  146  159
 100  118  105  128  111  122
 100  120  102  114  104  103
```

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado

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`)	El diseño tiene algunas deficiencias pero la ejecución es correcta (`1pts`).	El diseño es deficiente y 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 sintaxis que afectan el resultado (`0.5pts`)
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`2pts`)	El código es de buen performance durante la ejecución (`1.5pts`)	El código no está optimizado pero la ejecución no es deficiente(`1pts`)	El código no está optimizado y la ejecución es deficiente (`0pts`)

2. (6 points) **Multiplicación de los menores**
Escribir un programa que utilizando vectores, lea desde el teclado un vector de números enteros grandes de tamaño N, y reemplace cada valor del vector con el producto de aquellos valores que sean menores al valor a reemplazar, si el valor es el menor de todos (mínimo) no reemplazarlo.

Caso de uso #1
Input:

```
6
1 5 3 2 4 4
```

Output:

```
1 96 2 1 6 6
```

Caso de uso #2
Input:

```
20
2 3 1 2 3
4 1 2 5 6
7 1 9 5 1
6 4 5 8 7
```

Output:

```
1 8 1 1 8
72 1 1 1152 144000
5184000 1 2032128000 1152 1
144000 72 1152 254016000 5184000
```

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado

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`)	El diseño tiene algunas deficiencias pero la ejecución es correcta (`1pts`).	El diseño es deficiente y 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 sintaxis que afectan el resultado (`0.5pts`)
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`2pts`)	El código es de buen performance durante la ejecución (`1.5pts`)	El código no está optimizado pero la ejecución no es deficiente(`1pts`)	El código no está optimizado y la ejecución es deficiente (`0pts`)

3. (7 points) **Factorización de polinomio**
Un monomio es una expresión que sigue la siguiente forma: ax^b donde: a es el coeficiente y b el grado. Definir una clase que represente un monomio *classmonomio_t* y escribir un programa que permita generar un vector de monomios de tamaño N que lea el coeficiente y el grado de cada uno de los N monomio desde el teclado. El programa deberá generar un nuevo vector que almacene la factorización de los monomios de modo que solo se almacene un monomio por grado. El resultado deberá ser mostrado de modo que los monomios se muestren ordenados de menor grado a mayor grado.
Caso de uso #1
Input:

```
5
2 3
1 2
4 3
7 2
4 5
```

Output:

```
8x^2 6x^3 4x^5
```

Explicación: Utilizando el vector original de monomios se genera un nuevo vector que para este ejemplo finalmente tendrá 3 monomios esto debido a que solo existen 3 exponentes (2, 3, 5) y cuyos coeficientes serian (6=2+4, 8=1+7, 5) los cuales se muestran como un polinomio donde los monomios se muestran ordenados por el grado (2, 3, 5).

Caso de uso #2
Input:

```
6
10 5
1 1
2 2
3 4
2 1
8 5
```

Output:

```
3x^1 2x^2 3x^4 18x^5
```

Se aconseja mostrar algunos ejemplos sobre la ejecución correcta del código.

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado

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`)	El diseño tiene algunas deficiencias pero la ejecución es correcta (`1pts`).	El diseño es deficiente y 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 código es de buen performance durante la ejecución (`1.5pts`)	El código no está optimizado pero la ejecución no es deficiente(`1pts`)	El código no está optimizado y la ejecución es deficiente (`0pts`)