UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2
Pregrado
2022-2
Profesora: María Hilda Bermejo
Lab 2.06

Indicaciones específicas:

- Esta evaluación contiene 11 páginas (incluyendo esta página) con 3 preguntas. El total de puntos son 20.
- Esta prohibido el uso de celulares y audífonos
- Crea una carpeta llamada: PC2
- Luego crea una subcarpeta para cada pregunta:
  1. `p1`
  2. `p2`
  3. `p3`
- Una vez que termines de realizar las 3 preguntas:
  1. `Elimina el directorio cMake-Buil-debug de cada pregunta`
  2. `Comprime la carpeta PC2 usando el winzip y obtendrás el archivo PC2.zip`
- Sube el archivo **PC2.ZIP** al gradescoupe a www.gradescope.com.
- **Recuerda que solo se calificará si has enviado en el formato indicado.**

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) **Evalua matrices dinámicas**

Realice un programa que permita leer como dato el orden de una matriz cuadrada. El programa debe generar una matriz dinámica con valores aleatorios cuyos valores vayan desde 1 al 9, excepto en las posiciones que conforman la diagonal mayor, en donde se colocará el 0.
Luego el programa, formará una segunda matriz dinámica del mismo tamaño que la primera, pero en donde los elementos del triángulo inferior a la diagonal se encuentren ahora en el triángulo superior de la diagonal y visceversa. El programa imprimirá ambas matrices.

Para asignar el puntaje a esta pregunta es absolutamente necesario que:

- Valide el tamaño de la matriz, el número de filas y columnsa debe ser mayor a 4.
- Trabaje con matrices dinámicas
- **Cree la segunda matriz y no solo imprima los valores.**
- Imprima ambas matrices

Algunos ejemplos de diálogo de este programa serían:

Listing 1: Ejemplo 1

```
N [mayor a 4 ] = 3
N [mayor a 4 ] = 4
N [mayor a 4 ] = 6

Matriz 1

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

Matriz 2

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

Listing 2: Ejemplo 2

```
N [mayor a 4 ] = 5

Matriz 1

     0      2      8      1      8
     6      0      8      2      4
     7      2      0      6      5
     6      8      6      0      5
     7      1      8      2      0

Matriz 2

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

Listing 3: Ejemplo 3

```
N [mayor a 4 ] = 8

Matriz 1

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

Matriz 2

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

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`)

2. (6 points) **Vectores**

Realice un programa que lea el orden de una matriz, la llene con valores aleatorios entre 1 y 50 y luego forme un vector con los datos del borde de la matriz. Para formar el vector se debe recorrer el borde de la matriz en sentido horario.

El programa debe:

- Verificar que el orden de la matriz tenga al menos valores mayores a 2.
- La matriz que cree debe ser dinámica. No olvide liberar el espacio de memoria
- El programa debe formar el vector, no basta con imprimir los datos.
- Para asignar el puntaje a esta pregunta es **necesario que utilice un una matriz dinámica y un vector**

Listing 4: Ejemplo

```
Filas [mayor a 2] : 2
Columnas [mayor a 2 : 1
Filas [mayor a 2] : 4
Columnas [mayor a 2 : 6


    34     37     28     16     44     36
    37     43     50     22     13     28
    41     10     14     27     41     27
    23     37     12     19     18     30

Vector

  34   37   28   16   44   36   28   27   30   18   19   12   37   23   41
       37
Process finished with exit code 0
```

Listing 5: Ejemplo

```
Filas [mayor a 2] : 3
Columnas [mayor a 2 : 3


    34     37     28
    16     44     36
    37     43     50

Vector

  34   37   28   36   50   43   37   16
```

Listing 6: Ejemplo

```
Filas [mayor a 2] : 7
Columnas [mayor a 2 : 10



    34     37     28     16     44     36     37     43     50     22
    13     28     41     10     14     27     41     27     23     37
    12     19     18     30     33     31     13     24     18     36
    30      3     23      9     20     18     44      7     12     43
    30     24     22     20     35     38     49     25     16     21
    14     27     42     31      7     24     13     21     47     32
     6     26     35     28     37      6     47     30     14      8

Vector

  34   37   28   16   44   36   37   43   50   22   37   36   43   21   32
         8   14   30   47    6   37   28   35   26    6   14   30   30
      12   13
```

Listing 7: Ejemplo

```
Filas [mayor a 2] : 8
Columnas [mayor a 2 : 4



    34     37     28     16
    44     36     37     43
    50     22     13     28
    41     10     14     27
    41     27     23     37
    12     19     18     30
    33     31     13     24
    18     36     30      3

Vector

  34   37   28   16   43   28   27   37   30   24    3   30   36   18   33
       12   41   41   50   44
```

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`)

3. (7 points) **POO**

Desarrolle un programa orientado a objetos, que permita leer como dato un número entero y el programa a través de los servicios que ofrece la clase CArreglo, permita:

- Generar n números al azar cuyos valores estan entre 1 y 50 y los almacene en un arreglo dinámico de números enteros.
- Imprima el arreglo
- Halle e imprima la suma
- Halle e imprima el número mayor
- Halle e imprima el número menor
- Cuente el número de números primos que se encuentran en el arreglo. Si se ha generado números repetidos en el arreglo original, estos números entran en el conteo
- Haga un buen uso de la memoria
- **Para asignar el puntaje a esta pregunta es necesario que haya realizado un programa orientado a objetos. Es decir utilice clases y objetos.**

Listing 8: Ejemplo 4

```
 Numero de elementos : 6

Elementos del arreglo
   34   37   28   16   44   36

La suma de todos los elementos es: 195
El mayor elementos es: 44
El menor elemento es : 16
En el arreglo hay 1 numeros primos
```

Listing 9: Ejemplo 4

```
Numero de elementos : 23

Elementos del arreglo
   34   37   28   16   44   36   37   43   50   22   13   28
         41   10   14   27   41   27   23   37   12   19
      18

La suma de todos los elementos es: 657
El mayor elementos es: 50
El menor elemento es : 10
En el arreglo hay 9 numeros primos
```

Listing 10: Ejemplo 4

```
Numero de elementos : 30

Elementos del arreglo
   34    37    28    16    44    36    37    43    50    22    13    28
         41    10    14    27    41    27    23    37    12    19
      18    30    33    31    13    24    18    36

La suma de todos los elementos es: 842
El mayor elementos es: 50
El menor elemento es : 10
En el arreglo hay 11 numeros primos
```

A continuación se muestra parte del codigo que Ud. debe completar, de tal manera que el programa realice lo que se pide.

Listing 11: Ejemplo

```
#include "CArreglo.h"
#include <ctime>
#include <cstdlib>

using namespace std;

int main()
{ int n;

  srand(time(nullptr));
 cout << "Numero␣de␣elementos␣:␣";
 cin >> n;
 CArreglo a(n);
 a.imprimir();
 cout << "\n\n";
 cout << "La␣suma␣de␣todos␣los␣elementos␣es:␣" << a.suma()
    << "\n";
 cout << "El␣mayor␣elementos␣es:␣" << a.mayor() << "\n";
 cout << "El␣menor␣elemento␣es␣:␣" << a.menor() << "\n";
 cout << "En␣el␣arreglo␣hay␣" << a.contarPrimos() << "␣
    numeros␣primos\n";
  return 0;
}
```

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`)