UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2
Pregrado
2021-II
Profesor: Jorge Villavicencio
Lab: 1.07

Indicaciones específicas:

- Esta evaluación contiene 13 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 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) **Suma de diagonal principal de una matriz cuadrada**
Tema a evaluar: Matrices

Desarrollar un programa que permite sumar la diagonal principal de una matriz cuadrada aleatoria de numeros enteros.

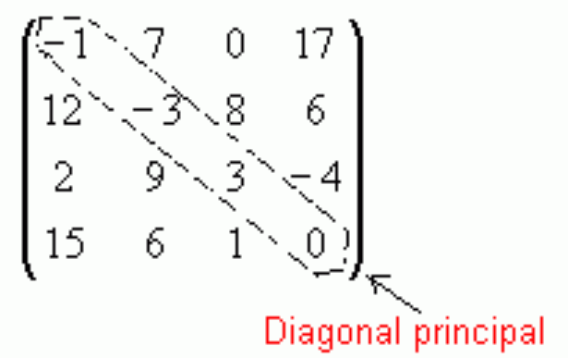

Figure 1: Diagonal principal de una matriz cuadrada.

Una matriz cuadrada se caracteriza porque su cantidad de filas es igual a la cantidad de columnas.

Para eso su programa debe ejecutar lo siguiente:

- El usuario debe ingresar la cantidad de filas y columnas para el programa.
- El valor de la filas y columnas tienen que ser mayor igual a 2, en caso que el usuario ingrese un número menor al 2 se debe volver a solicitar.
- Generación de la matriz de enteros con números aleatorios entre 1 a 100.
- Mostrar la matriz generada al usuario.
- Indicar la sumatoria de la diagonal principal de la matriz.
- Liberar la memoria dinámica utilizada.

IMPORTANTE: En este ejercicio no puede utilizar librerias. Para resolver el ejercicio debe utilizar matrices dinámicas.

Listing 1: Ejemplo de resultado 1

```
Filas=Columnas:5
81      53      9       65      60
13      27      81      23      10
83      39      93      91      94
61      5       40      82      44
48      67      96      54      24
Sumatoria de diagonal principal:307
```

Listing 2: Ejemplo de resultado 2

```
Filas=Columnas:1
Filas=Columnas:-1
Filas=Columnas:0
Filas=Columnas:8
13      96      6       66      22      99      48      16

31      86      74      55      76      1       83      18

11      50      73      92      98      24      39      25

81      24      80      46      99      88      31      31

27      69      35      64      36      13      97      60

46      3       36      69      1       84      51      56

90      1       69      20      63      69      4       77

50      87      82      71      57      41      24      46

Sumatoria de diagonal principal:388
```

Listing 3: Ejemplo de resultado 3

```
Filas=Columnas:0
Filas=Columnas:6
23      89      87      57      8       64
56      65      77      11      43      97
62      2       81      47      30      67
42      87      51      50      36      90
1       6       32      48      89      41
49      49      33      20      79      88
Sumatoria de diagonal principal:396
```

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) **Creación de listas consonantes, vocales y otros.**
Tema a evaluar: Vectores

Diseñe e implemente un programa que permita ingresar varias palabras hasta que el usuario ingrese *SALIR*. Con el listado de palabras se deben crear 3 listas. La primera lista con todas las palabras que empiezan con vocal **('a', 'e', 'i', 'o' y 'u')**, la segunda lista con todas las palabras que empiezan con consonantes **('b','c','d' ... 'x','y','z')**, y la última lista con las palabras restantes que no esten en la primera y segunda lista. Finalmente, imprimimos las tres listas. **No considerar la palabra SALIR en ninguna lista.**

IMPORTANTE: En este ejercicio puede utilizar la biblioteca String, Vector. **Código ASCII de la letra 'b' es 98 y de la letra 'z' es 122.** Considerar solamente palabras que comiencen en minúsculas.

Listing 4: Ejemplo de resultado 1

```
Palabra: 1
Palabra: miraflores
Palabra: barranco
Palabra: palermo
Palabra: inmaduro
Palabra: oregano
Palabra: uuuuu
Palabra: elefefanta
Palabra: 2
Palabra: 3
Palabra: SALIR
La primera lista: [inmaduro, oregano, uuuuu, elefefanta, ]
La segunda lista: [miraflores, barranco, palermo, ]
La tercera lista: [1, 2, 3, ]
```

Listing 5: Ejemplo de resultado 2

```
Palabra: 12
Palabra: 1luis
Palabra: anamaria
Palabra: felipe
Palabra: pimentel
Palabra: enamorados
Palabra: opulencia
Palabra: queen
Palabra: tambor
Palabra: iiiii
Palabra: SALIR
La primera lista: [anamaria, enamorados, opulencia, iiiii, ]
La segunda lista: [felipe, pimentel, queen, tambor, ]
La tercera lista: [12, 1luis, ]
```

Listing 6: Ejemplo de resultado 3

```
Palabra: ana
Palabra: oso
Palabra: pelicano
Palabra: iguana
Palabra: amor
Palabra: mentira
Palabra: ilusion
Palabra: SALIR
La primera lista: [ana, oso, iguana, amor, ilusion, ]
La segunda lista: [pelicano, mentira, ]
La tercera lista: []
```

Listing 7: Ejemplo de resultado 4

```
Palabra: SALIR
La primera lista: []
La segunda lista: []
La tercera lista: []
```

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) **Promedio de notas de alumno UTEC.**
Tema a evaluar: Clases y Objetos

Se te solicita construir un software que permita ingresar los datos básicos(nombre, apellido paterno y apellido materno) de un alumno de UTEC e indicar cuales son las notas que ha acumulado en el curso de Programación II.

Para cumplir con este requerimiento debes crear un programa que permita solicitar los datos de 'n' cantidad de notas y almacenar esos datos en el atributo **v_notas** de la clase CAlumno. La clase tiene tiene los siguientes atributos y métodos especificados:

CAlumno
nombre: string apellido_paterno: string apellido_materno: string cant_notas: int v_notas: vector<float>
CAlumno() CAlumno(_nombre, _apellido_paterno,_apellido_materno) ~CCAlumno() getNombre() getApellido_paterno() getApellido_materno() getCant_notas() setNombre(_nombre) setApellido_paterno(_apellido_paterno) setApellido_materno(_apellido_materno) setCant_notas(_cant_notas) agregarNota(_nota) mostrarNotas() promedioNotas()

Además, las notas se van almacenar en un vector de tipo float y se agregan utilizando el método **agregarNota(_nota)**.

Para terminar el programa debe imprimir el listado de notas del alumno y su promedio utilizando el método **promedioNotas()**.

IMPORTANTE: En este ejercicio puede utilizar la biblioteca String y Vector. Asumir que el usuario ingresará notas en el rango de 0 a 20.

A continuación se muestra algunos ejemplos de la ejecución correcta del código:

Listing 8: Ejemplo de resultado 1

```
Luis
Vidal
Valencia
Ingresar cantidad de notas: 4
Nota 1: 10.4
Nota 2: 12
Nota 3: 15
Nota 4: 18

Listado de notas del alumno: Luis Vidal Valencia
1) 10.4
2) 12
3) 15
4) 18

Promedio: 13.85
```

Listing 9: Ejemplo de resultado 2

```
Jorge
Villavicencio
Antunez
Ingresar cantidad de notas: 12
Nota 1: 12
Nota 2: 10
Nota 3: 4
Nota 4: 18
Nota 5: 15
Nota 6: 6
Nota 7: 0
Nota 8: 14
Nota 9: 18
Nota 10: 9
Nota 11: 12
Nota 12: 19

Listado de notas del alumno: Jorge Villavicencio Antunez
1) 12
2) 10
3) 4
4) 18
5) 15
6) 6
7) 0
8) 14
9) 18
10) 9
11) 12
12) 19

Promedio: 11.4167
```

Listing 10: Ejemplo de resultado 3

```
Milagros
Zamora
Retis
Ingresar cantidad de notas: 6
Nota 1: 20
Nota 2: 12
Nota 3: 9
Nota 4: 12
Nota 5: 18.5
Nota 6: 10.5

Listado de notas del alumno: Milagros Zamora Retis
1) 20
2) 12
3) 9
4) 12
5) 18.5
6) 10.5

Promedio: 13.6667
```

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`)