UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2
Pregrado
2021-II
Profesor: Jorge Villavicencio
Lab: 1.01

Indicaciones específicas:

- Esta evaluación contiene 10 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 el borde de los elementos de una matriz**
Tema a evaluar: Matrices

Desarrollar un programa que permite sumar los bordes de una matriz aleatoria de numeros enteros. 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 los bordes 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:1
Filas:0
Filas:2
Columnas:6
58      7       53      83      75      87
10      59      31      93      28      83
Sumatoria de borde:667
```

Listing 2: Ejemplo de resultado 2

```
Filas:4
Columnas:4
72      94      80      37
50      97      63      72
87      99      75      22
51      56      79      49
Sumatoria de borde:749
```

Listing 3: Ejemplo de resultado 3

```
Filas:4
Columnas:1
Columnas:-4
Columnas:5
5       84      9       16      50
77      52      43      13      71
59      34      47      7       85
36      87      69      11      50
Sumatoria de borde:709
```

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 de vocales.**
Tema a evaluar: Vectores

Diseñe e implemente un programa que permita ingresar varias palabras hasta que el usuario ingrese *-1*. Con el listado de palabras se deben crear 3 listas. La primera lista con todas las palabras que empiezan con vocal abiertas **('a', 'e' y 'o')**, la segunda lista con todas las palabras que empiezan con vocal cerradas **('i' y 'u')**, y la última lista con las palabras restantes que no esten en la primera y segunda lista. Finalmente, imprimimos las tres listas. **No considerar el -1 en ninguna lista.**

IMPORTANTE: En este ejercicio puede utilizar la biblioteca String, Vector.

Listing 4: Ejemplo de resultado 1

```
Palabra: orca
Palabra: ana
Palabra: iman
Palabra: pelicano
Palabra: ulaula
Palabra: enano
Palabra: marciano
Palabra: -1
La primera lista: [orca, ana, enano, ]
La segunda lista: [iman, ulaula, ]
La tercera lista: [pelicano, marciano, ]
```

Listing 5: Ejemplo de resultado 2

```
Palabra: oso
Palabra: a
Palabra: e
Palabra: i
Palabra: o
Palabra: u
Palabra: furia
Palabra: desodorante
Palabra: televisor
Palabra: netflix
Palabra: imantado
Palabra: -1
La primera lista: [oso, a, e, o, ]
La segunda lista: [i, u, imantado, ]
La tercera lista: [furia, desodorante, televisor, netflix, ]
```

Listing 6: Ejemplo de resultado 3

```
Palabra: a
Palabra: e
Palabra: i
Palabra: -1
La primera lista: [a, e, ]
La segunda lista: [i, ]
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) **Listado de cursos de un profesor de 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 profesor de UTEC e indicar cuales son los cursos que está dictando en el año 2022-1.

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

CProfesor
nombre: string apellido_paterno: string apellido_materno: string cant_cursos: int v_cursos: vector<string>
CProfesor() CProfesor(_nombre, _apellido$_p$*aterno*, _*apellido*$_m$*aterno*) ~CProfesor() getNombre() getApellido_paterno() getApellido_materno() getCant_cursos() setNombre(_nombre) setApellido_paterno(_apellido_paterno) setApellido_materno(_apellido_materno) setCant_cursos(_cant_cursos) agregarCurso(_nombreCurso) mostrarCursos()

Además, los nombres de los curso se van almacenar en un vector de tipo string y se agregan utilizando el método **agregarCurso(_nombreCurso)**.

Para terminar el programa debe imprimir el listado de cursos del profesor y la cantidad de cursos que dicta en UTEC.

IMPORTANTE: En este ejercicio puede utilizar la biblioteca String y Vector.

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

Listing 7: Ejemplo de resultado 1

```
Luis
Vidal
Martinez
Ingresar cantidad de cursos: 3
Curso 1: Calculo
Curso 2: Programacion
Curso 3: Estadistica

Listado de cursos del profesor: Luis Vidal Martinez
1) Calculo
2) Programacion
3) Estadistica

Cantidad de cursos: 3
```

Listing 8: Ejemplo de resultado 2

```
Jorge
Villavicencio
Antunez
Ingresar cantidad de cursos: 4
Curso 1: Ingles
Curso 2: Programacion
Curso 3: Variables
Curso 4: Fisica

Listado de cursos del profesor: Jorge Villavicencio Antunez
1) Ingles
2) Programacion
3) Variables
4) Fisica

Cantidad de cursos: 4
```

Listing 9: Ejemplo de resultado 3

```
Jesus
Fiestas
Vidal
Ingresar cantidad de cursos: 6
Curso 1: Programacion
Curso 2: Integrales
Curso 3: Dise o
Curso 4: Electivo
Curso 5: Agil
Curso 6: Calculo

Listado de cursos del profesor: Jesus Fiestas Vidal
1) Programacion
2) Integrales
3) Dise o
4) Electivo
5) Agil
6) Calculo

Cantidad de cursos: 6
```

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`)