UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2
Pregrado
2021-II
Profesor: Jonathan Silva
Lab: 2.05

Indicaciones específicas:

- Esta evaluación contiene 9 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) **Buscar palabra horizontal**
Tema a evaluar: Matrices

Desarrollar un programa que permita buscar caracteres continuos de forma horizontal. Para eso su programa debe ejecutar lo siguiente:

- El usuario debe ingresar la cantidad de filas y columnas para el programa.
- Su programa debe generar una matriz de caracteres aleatorios con las dimensiones de filas y columnas indicadas por el usuario.
- Luego debe mostrar la matriz generada al usuario.
- El usuario indica los caracteres consecutivos que desea buscar.
- El programa muestra las posiciones de inicio y fin de la secuencia de caracteres.

Puede asumir que el usuario solo desea buscar caracteres de forma horizontal. Una vez encontrado los caracteres en la lista, debe imprimir las posiciones de inicio y fin de esos caracteres. Si no se encuentran los caracteres consecutivos se le informa al usuario que no se encontraron en la matriz.

IMPORTANTE: En este ejercicio no puede utilizar librerias. Para resolver el ejercicio debe utilizar matrices dinámicas. Recuerde que el caracter A y Z es el código 65 y 90 del estándar ASCII respectivamente.

Listing 1: Ejemplo de resultado 1

```
Ingrese las dimensiones de filas y columnas:
4
13
La matriz generada es:
P H Q G H U M E A Y L N L
F D X F I R C V S C X G G
B W K F N Q D U X W F N F
O Z V S R T K J P R E P G
Ingrese los caracteres a buscar:FNQD
Los caracteres FNQD se encuentran en las posiciones:
Inicio: 2,3
Fin: 2,6
```

Listing 2: Ejemplo de resultado 2

```
Ingrese las dimensiones de filas y columnas:
4
13
La matriz generada es:
P H Q G H U M E A Y L N L
F D X F I R C V S C X G G
B W K F N Q D U X W F N F
O Z V S R T K J P R E P G
Ingrese los caracteres a buscar:UTEC
Los caracteres UTEC no se encuentran en la matriz.
```

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) **Encontrar elementos iguales**
Tema a evaluar: Vectores

Desarrollar un programa que genere 2 vectores de caracteres a partir de 2 palabras ingresadas por el usuario.

Luego, tiene que buscar los caracteres iguales entre ambos vectores y en caso que no existan, informar que no se encontraron.

IMPORTANTE: En este ejercicio puede utilizar la biblioteca String, Vector y Algorithm. Para resolver el ejercicio debe utilizar vectores para realizar las comparaciones de caracteres.

Listing 3: Ejemplo de resultado 1

```
Ingrese la palabra 1:
barranco
Ingrese la palabra 2:
laborar

Los vectores creados son:
b, a, r, r, a, n, c, o,
l, a, b, o, r, a, r,
Las letras comunes son:
a,  b,  o,  r,
```

Listing 4: Ejemplo de resultado 2

```
Ingrese la palabra 1:
puerta
Ingrese la palabra 2:
sismo

Los vectores creados son:
p, u, e, r, t, a,
s, i, s, m, o,
No tienen letras comunes.
```

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) **Empresa de Taxi y Buses**
Tema a evaluar: Clases y Objetos

Un amigo te solicita ayuda para construir un software que permita recolectar los destinos más frecuentes para una empresa de Taxi y las estaciones más frecuentes para una empresa de Bus.

Para cumplir con este requerimiento debes crear un programa que permita solicitar los datos de 'n' cantidad de lugares y almacenar esos datos en una clase CTaxi y otra clase CBus según los atributos y métodos especificados a continuación:

CTaxi	**CBus**
nombre: string lugares: vector<string>	nombre: string lugares: vector<string>
CTaxi() CTaxi(_nombre, _destinos) ~CTaxi() getNombre() getCantidadLugares() getLugar(posicion)	CBus() CBus(_nombre, _estaciones) ~CBus() getNombre() getCantidadLugares() getLugar(posicion)

Además, acuerdan que para recibir los datos del usuarios van a utilizar una variable vector y un constructor que permita copiar los elementos del vector.

Para terminar el programa tiene que imprimir los lugares ingresados por el usuario con solo una función genérica que reciba ambas clases y permita imprimir los destinos de la empresa de taxi y las estaciones de la empresa de bus.

IMPORTANTE: En este ejercicio puede utilizar la biblioteca String y Vector. Note que tiene que leer strings de múltiples palabra con la sentencia getline.

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

Listing 5: Ejemplo de resultado 1

```
Ingrese el nombre de la empresa de taxi:
El rapido
Ingrese cantidad de destinos frecuentes:
4
Ingrese los 4 destinos frecuentes:
Aeropuerto Jorge Chavez
Catedral de Lima
Cerro San Cristobal
Playa Punta Hermosa
Ingrese el nombre de la empresa de bus:
La ruta
Ingrese cantidad de estaciones frecuentes:
4
Ingrese las 4 estaciones frecuentes:
Ica
Arequipa
Trujillo
Huancayo

El rapido tiene los siguientes lugares mas visitados:
1) Aeropuerto Jorge Chavez
2) Costa Verde
3) Cerro San Cristobal
4) Playa Punta Hermosa
La ruta tiene los siguientes lugares mas visitados:
1) Ica
2) Arequipa
3) Trujillo
4) Huancayo
```

Listing 6: Ejemplo de resultado 2

```
Ingrese el nombre de la empresa de taxi:
El rapido
Ingrese cantidad de destinos frecuentes:
1
Ingrese los 1 destinos frecuentes:
Aeropuerto Jorge Chavez
Ingrese el nombre de la empresa de bus:
La ruta
Ingrese cantidad de estaciones frecuentes:
2
Ingrese las 2 estaciones frecuentes:
Ica
Arequipa

El rapido tiene los siguientes lugares mas visitados:
1) Aeropuerto Jorge Chavez
La ruta tiene los siguientes lugares mas visitados:
1) Ica
2) Arequipa
```

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)