UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2
Pregrado
2021-II
Profesor:
Lab:

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) **Pregunta 1. Matrices con punteros.**
Se tiene una matriz donde las filas definen los usuarios de un sistema, y las columnas definen los permisos del usuario a un archivo en particular. La primera columna de la matriz consiste de los nombres de usuarios, y la primera fila consiste de los nombres de los archivos. Existen tres acciones a realizar en el sistemas: leer, escribir y borrar. Para simplificar las cosas, una entrada en la matriz es un String de tres caracteres, con los permisos respectivos para cada una de las accciones, representados por los caracteres: r, w, y d respectivamente. Si un usuario no tiene un permiso específico para un archivo, el caracter se representa con un guión: '-' Por ejemplo, si Jose tiene permisos de leer y borrar para el archivo "archivo1", la entrada correspondiente en la matriz será "r-d".

Se le pide implementar un programa interactivo que permita modificar los permisos de un usuario para un archivo dado. El programa debe preguntar si se quiere agregar o remover un permiso, luego pide el permiso a modificar y finalmente pide el nombre del usuario y el nombre del archivo. Puedes utilizar la matriz base que está en los ejemplos debajo. Además, se deben usar punteros para manipular los datos en la matriz. Tomar en cuenta que no se necesita hacer ninguna validación adicional sobre los datos ingresados Algunos ejemplos del programa serían:

Listing 1: Ejemplo 1

```
Matriz inicial
        misNotas.txt    tarea1.pdf     ataque.sh
Jose      r-d             -w-             r--
Pedro     -wd             rwd             rw-
Luis      --d             -wd             ---

Desea agregar o remover un permiso? R
Elija el tipo de permiso: r
Elija un usuario: Jose
Elija un archivo: misNotas.txt

Desea agregar o remover un permiso? F

Process finished with exit code 0

Matriz final
        misNotas.txt    tarea1.pdf     ataque.sh
Jose      --d             -w-             r--
Pedro     -wd             rwd             rw-
Luis      --d             -wd             ---
```

Listing 2: Ejemplo 2

```
Matriz inicial
        misNotas.txt    tarea1.pdf     ataque.sh
Jose      r-d             -w-             r--
Pedro     -wd             rwd             rw-
Luis      --d             -wd             ---

Desea agregar o remover un permiso? A
Elija el tipo de permiso: d
Elija un usuario: Luis
Elija un archivo: ataque.sh
Desea agregar o remover un permiso? R
Elija el tipo de permiso: w
Elija un usuario: Pedro
Elija un archivo: tarea1.pdf

Desea agregar o remover un permiso? F

Matriz final
        misNotas.txt    tarea1.pdf     ataque.sh
Jose      --d             -w-             r--
Pedro     -wd             r-d             rw-
Luis      --d             -wd             --d
```

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) Pregunta 2. Vectores.
Se tienen dos vectores con números enteros. Implementa un programa que haga lo siguiente:

- **Cree un nuevo vector de la unión de los elementos de los vectores iniciales, sin repeticiones.**
- **Crear dos nuevos vectores de modo que tienen los elementos del primero y el segundo respectivamente, pero sin los elementos que existen en ambos al mismo tiempo. Notar que un vector puede tener elementos repetidos, se deben eliminar todas las copias de los elementos que cumplan la condición.**

Hint: Probablemente te sirva crear una función para buscar elementos en un vector, o usar la función find del header "algorithm", también podría serte útil crear una pequeña función para imprimir valores del vector
Algunos ejemplos de la ejecución de este programa:

Listing 3: Ejemplo 1

```
vector1: {1,2,1,3,4,5}
vector2: {1,6,7,8,9}

vector3: {1,2,3,4,5,6,7,8,9}
vector1 modificado: {2,3,4,5}
vector2 modificado: {6,7,8,9}
```

Listing 4: Ejemplo 2

```
vector1: {2,2,3,3}
vector2: {2,3,4,5}

vector3: {2,3,4,5}
vector1 modificado: {}
vector2 modificado: {4,5}
```

Listing 5: Ejemplo 3

```
vector1: {2,2,3,3}
vector2: {2,3}

vector3: {2,3}
vector1 modificado: {}
vector2 modificado: {}
```

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) Pregunta 3. Clases y Objetos.
Un profesor de UTEC quiere automatizar el proceso para ajustar la nota de sus alumnos. Para ello se pide crear una clase CAlumno, que tenga las siguientes propiedades: nombre, edad, nota. Se pide implementar una función que reciba una lista de CAlumno*, ya sea un vector o un array, y calcule el promedio de los alumnos. Luego, se modifica las notas de los alumnos de la siguiente manera:

- **Si el promedio es menor a 11, todos los alumnos reciben 20% más de nota.**
- **Si el promedio está entre 11 y 15, todos los alumnos reciben 10% más de nota.**

Finalmente, realizar un reporte de los alumnos con nombre, edad y nota, separando los reprobados y los aprobados. Se deben implementar los metodos y funciones necesarias, incluyendo el constructor, los getters y los setters.

Un ejemplo de creación de alumnos, el primer numero es la edad y el segundo es la nota; y el resultado del programa:

Listing 6: Ejemplo 1

```
#creacion de alumnos
vector<CAlumno*> alumnos = {new CAlumno("Pedro", 18, 15),
                            new CAlumno("Pablo", 17, 4),
                            new CAlumno("Maria", 18, 10),
                            new CAlumno("Juana", 19, 5),
                            new CAlumno("Tomas", 20, 12)};

#resultado de ejecucion del programa
==================
Promedio del curso: 9.2
==================
Alumnos reprobados
==================
Nombre: Pablo Edad: 17 Nota: 4.8
Nombre: Juana Edad: 19 Nota: 6
==================
Alumnos aprobados
==================
Nombre: Pedro Edad: 18 Nota: 18
Nombre: Maria Edad: 18 Nota: 12
Nombre: Tomas Edad: 20 Nota: 14.4

Process finished with exit code 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)