UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 3
Pregrado
2022-II
Profesor Wilder Nina Choquehuayta
Lab 1.02

Indicaciones específicas:

- Esta evaluación contiene 5 páginas (incluyendo esta página) con 1 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`
- 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 y de matemáticas apropiadas para la disciplina. (Evaluar)

  Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución.(Usar)

  Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (Usar)

- Para los alumnos de las carreras de Ingeniería

  Capacidad de aplicar conocimientos de matemáticas (nivel 3)

  Capacidad de aplicar conocimientos de ingeniería(nivel 2)

  Capacidad para diseñar un sistema, un componente o un proceso para satisfacer las necesidades deseadas dentro de restricciones realistas (nivel 2)

Calificación:

Tabla de puntos (sólo para uso del professor)

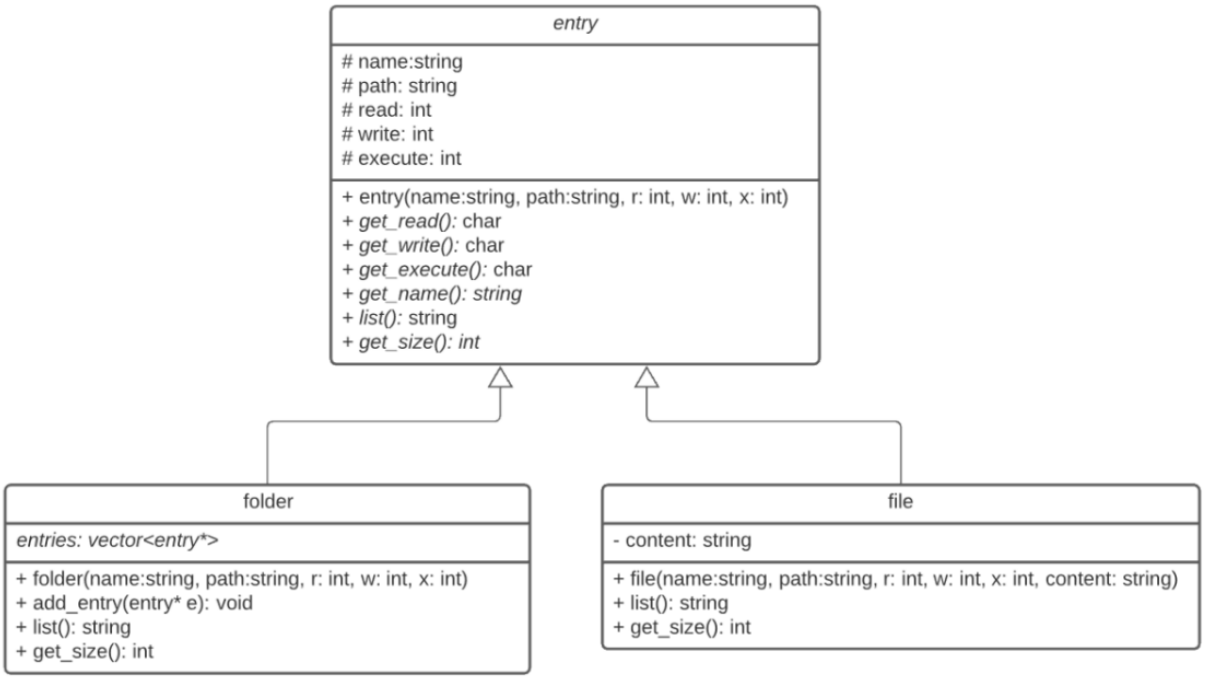
Question	Points	Score
1	20	
Total:	20	

1. (20 points) Escribir un programa que utilizando la clase **entry** y sus clases derivadas **folder** y **file** genere un objeto principal (root) del tipo folder, que almacene n files y folders, ingresando su tipo (D=Folder y F=File) , nombre, acceso a lectura (0=no acceso y 1=acceso), acceso a escritura (0=no acceso y 1=acceso), acceso a ejecucion (0=no acceso y 1=acceso) y en el caso de **file** el contenido, para ello debe utilizarse el operador **add** de **root** , los metodos **get_read()** devolvera **'R'** si tiene acceso y **'-'** si no tiene acceso, **get_write()** devolvera **'W'** si tiene acceso y **'-'** si no tiene acceso y **get_execute()** devolvera **'X'** si tiene acceso y **'-'** si no tiene acceso, el método **get_size()** devolvera **0** en caso de **folder** y el tamaño del atributo **content** en caso de **file**.

   El programa debe de retornar la lista de archivos y folders de **root** utilizando el operador sobrecargado:

```
ostream& operator<<(ostream& out, folder f); // opcion 1
ostream& operator<<(ostream& out, entry* f); // opcion 2
```

Figure 1: Diagrama de Clases



   Para el resolver el ejercicio debera utilizar Herencia, Polimorfismo y Sobrecarga de Operadores.

Listing 1: Entrada

```
D root 1 1 0
7
D folder1 1 1 0
F file1 1 1 0 contenidoX
F file2 1 1 1 contenidoY
F file3 1 1 0 contenidoZ
D folder2 1 0 0
D folder3 0 0 0
D folder4 0 0 0
```

Listing 2: Salida

```
RW- root
RW- root/folder1
RW- root/file1 contenidoX
RWX root/file2 contenidoY
RW- root/file3 contenidoZ
R-- root/folder2
--- root/folder3
--- root/folder4
```

La rúbrica para esta pregunta es:

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (`10pts`)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (`5pts`)	La ejecución es correcta (`3pts`).	La ejecución no es correcta (`0pts`)
Sintaxis	No existen errores sintácticos o de compilación (`5pts`)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (`3pts`).	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`).
Optimización	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`5pts`)	El codigo es de buen performance durante la ejecución (`3pts`).	El codigo no está optimizado pero la ejecución no es deficiente(`1pts`).	El codigo no está optimizado y la ejecución es deficiente (`0pts`).