Tercer Parcial

7 de mayo de 2020

Indicaciones generales

1. Este es un examen **individual** con una duración de **120 minutos: de 7:00 a 9:00 am**.
2. En `e-aulas` puede acceder a las diapositivas y a la sección correspondiente a este parcial.
3. Solamente será posible tener acceso a `e-aulas.urosario.edu.co` y a los sitios web correspondientes a la documentación de C++ dispuestos por el profesor.
4. Maletas, morrales, bolsos, etc. deben estar ubicados al frente del salón.
5. Celulares y otros dispositivos electrónicos deben estar apagados y ser guardados dentro de las maletas antes de ser ubicadas en su respectiva posición.
6. El estudiante no debe intentar ocultar ningún código que no sea propio en la solución a la actividad.
7. El estudiante solo podrá disponer de hojas en blanco como borrador de apuntes (opcional).
8. El estudiante puede tener una hoja manuscrita de resumen (opcional). Esta hoja debe estar marcada con nombre completo.
9. `e-aulas` se cerrará a la hora en punto acordada. La solución de la actividad debe ser subida antes de esta hora. El material entregado a través de `e-aulas` será calificado tal como está. Si ningún tipo de material es entregado por este medio, la nota de la evaluación será 0.0. **Se aconseja subir a e-aulas versiones parciales de la solución a la actividad.**
10. Todas las evaluaciones serán realizadas en el sistema operativo GNU/Linux.
11. Todas las entregas están sujetas a herramientas automatizadas de detección de plagio en códigos.
12. La evaluación debe presentarse exclusivamente en uno de los computadores ubicados en el salón de clase y a la hora acordada. Presentar la evaluación desde otro dispositivo o en otro horario diferente al estipulado es causa de anulación.
13. **Cualquier incumplimiento de lo anterior conlleva la anulación del examen.**
14. Las respuestas deben estar totalmente justificadas.
15. **Entrega**: archivos con extensión `.txt`, `.cpp` o `.hpp` según el caso, conteniendo la demostración o el código. Nombre los archivos como `pY.Z`, con `Y = 1,2,3` y `Z = txt, cpp, hpp`. Comprima su código y demás archivos en *un único* archivo `parcial.zip` y súbalo a `e-aulas`. **Importante:** no use acentos ni deje espacios en los nombres de los archivos que cree.

En el desarrollo del examen, no olvide usar la plantilla para cada ejercicio.
Las implementaciones deben ejecutarse sin errores usando las funciones `main()` de cada plantilla.

1. [50 ptos.] En una estación de trenes se tiene un tren como una serie de vagones enganchados a una locomotora: uno después del otro en una sola vía férrea. Un trabajador de la estación se encuentra llevando un registro de cuál es el número de pasajeros en el primer vagón enganchado a la locomotora y un registro de cuántos vagones tienen un rango específico de pasajeros. El trabajador lleva este registro en forma de una lista simplemente enlazada, donde la cabeza de la lista corresponde al último vagón enganchado al tren; es decir, el más alejado de la locomotora. Por lo tanto, el primer vagón enganchado al tren, el más cercano a la locomotora, corresponde al elemento más alejado de la cabeza de la lista enlazada.

   El registro de vagones implementado para esta tarea tiene una funcionalidad mínima, y la información guardada en este registro corresponde al número de pasajeros en cada vagón enganchado al tren. Con el objetivo de lograr la funcionalidad que el trabajador de la estación de trenes necesita para llevar a cabo exitosamente su labor, incorpore los siguientes métodos a la implementación básica de una lista simplemente enlazada (*singly linked list*).

   a) [25 ptos.] Un método que ubique el vagón que se encuentra inmediatamente después de la locomotora y retorne el número actual de pasajeros.

   ```
   template <typename T>
   T back() const;
   ```

   Note que cuando el tren tiene solamente un vagón, los métodos `back` y `front` deben retornar el mismo número de pasajeros. Si el tren no tiene vagones se debe reportar un error y terminar la ejecución del programa.

   b) [25 ptos.] Un método que revise todos los vagones del tren y cuente cuántos vagones tienen un número de pasajeros en el rango `[begin, end]`; note que el intervalo es **cerrado**:

   ```
   template <typename T>
   int range_counting(T begin, T end) const;
   ```

   Este rango es determinado por el trabajador y se debe cumplir siempre que `begin <= end`, en caso que esto no se cumpla el método debe retornar -1. En caso que el tren no tenga vagones debe retornar 0.

   Pruebe la correctitud de la funcionalidad implementada creando un registro de pasajeros en cada vagón; es decir, una lista de números enteros. Invoque estos métodos en diferentes circunstancias: (*i*) un tren sin vagones enganchados, (*ii*) un tren con un solo vagón enganchado, y (*iii*) un tren con más de un vagón enganchado a la locomotora.

2. [50 ptos.] [*Binary search tree with values.*] Los algoritmos de inserción, borrado y búsqueda sobre un árbol binario de búsqueda se basan en la manera en la que sus elementos están ordenados, de acuerdo a su llave o `key`. La llave es, entonces, el medio para encontrar información en el árbol, mas no la información en sí misma que se desea guardar.

   El presente ejercicio consiste en iplementar, en un árbol que contiene el nuevo atributo `value`, y cuyos nodos son de la forma

```
tem plate <typename keyType, typename valType>
struct BSTNode {
    keyType key;
    valType value;
    BSTNode<keyType, valType> *left;
    BSTNode<keyType, valType> *right;
    BSTNode<keyType, valType> *parent;
};
```

   los siguientes métodos:

   a) [25 ptos.] `void insert(keyType k, valType v)`: Método que inserta en el árbol un nodo con llave `k` y valor `v` si la llave no se encuentra actualmente en el árbol. De lo contrario reemplaza el valor actual asociado a `k` por `v`.

   b) [25 ptos.] `valType get(keyType k)`: Método que retorna el valor asociado a la llave `k`. Si la llave no se encuentra en la estructura, arroja un error estándar `runtime_error`.

   Usando el archivo `main.cpp`, asegúrese que el programa compila después de realizar los cambios a la plantilla.