Segundo Parcial

16 de noviembre de 2019

Indicaciones generales

1. Este es un examen **individual** con una duración de **120 minutos: de 11:00 a 13:00**.
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.] [*Reverse linked list.*] Considere la implementación de una lista simplemente enlazada (*singly-linked list*). Extienda esta estructura de datos de forma que incluya el siguiente método público:

```
void reverse();
```

   Su solución puede ser tanto recursiva como iterativa. Este método invierte el orden de los elementos en la lista enlazada SIN CREAR UNA LISTA NUEVA O AÑADIR/QUITAR NODOS A LA LISTA EXISTENTE. Por ejemplo, suponga que tiene una lista `List<char> mylst` con el contenido `'M' -> 'A' -> 'C' -> 'K'`. El resultado de invocar `mylst.reverse()` es que el nuevo contenido de la lista es `'K' -> 'C' -> 'A' -> 'M'`.

2. [50 ptos.] Agregue a la implementación e interfaz de un árbol binario de búsqueda las siguientes rutinas que no modifican el estado del árbol. Considere la estructura de datos árbol binario de búsqueda (*binary search tree*). Implemente como métodos privado y público, respectivamente, los siguientes predicados:

```
bool check_if_bst(bstNode *root) const; // private
bool check_if_bst() const;              // public
```

   Estos verifican si el árbol binario, que tiene como raíz a `root`, satisface la propiedad de árbol binario de búsqueda. Su implementación debe tomar tiempo lineal en el número de nodos del árbol binario. El método público solamente debe invocar al método privado con los parámetros apropiados.