Primer Parcial

5 de septiembre de 2020

Indicaciones generales

1. Este es un examen **individual** con una duración de **120 minutos: de 10:00 am a 12:00 m**.
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. [30 ptos.] Suponga dos conjuntos de enteros `set<int>`. Utilizando un tercer `set` para registrar el resultado y solamente los métodos exportados por esta librería, implemente la operación de intersección de conjuntos. El prototipo de la función que realiza la operación debe ser

   ```
   1 set<int> intsect(const set<int> &a, const set<int> &b);
   ```

   Note que los vectores son pasados como referencias constantes a la función. Es decir, los conjuntos `a` y `b` no pueden ser modificados dentro de la función.

   Recuerde que en teoría de conjuntos, la intersección de dos (o más) conjuntos es una operación que resulta en otro conjunto que contiene los elementos comunes a los conjuntos de partida. Por ejemplo, dados dos conjuntos `a` y `b`

   ```
   a = {5, 4, 3, 2, 1}
   b = {1, 3, 9, 7}
   ```

   su intersección es el conjunto `c` cuyo contenido

   ```
   c = {1, 3}
   ```

   corresponde a los elementos que están en el conjunto `a` *y* a los que están en `b`.

2. [30 ptos.] Considere un vector de cadenas de caracteres (`string`) que puede contener elementos duplicados. Usando `map` construya una función que retorna la cadena de caracteres que más se repite en el contenedor `vector`. La firma de la función construida debe ser de la forma

   ```
   1 string max_reps(const vector<string> &vec);
   ```

   donde `vec` es el vector que contiene los elementos del tipo `string`. Como ejemplo, considere el vector

   ```
   {"Z", "K", "g", "e", "K", "e", "a", "K"}
   ```

   note que los elementos que se repiten son `"K"` 3 veces y `"e"` 2 veces; sin embargo, el elemento que más se repite es `"K"`. Por lo tanto, la función debe retornar `"K"`.

3. [40 ptos.] Implemente la función

   ```
   1 void sort_even(vector<int> &vec);
   ```

   que recibe *por referencia* un vector de números enteros. Usando una variación de ordenamiento por selección, la función debe ordenar de manera ascendente de izquierda a derecha únicamente los elementos pares, dejando los elementos impares en sus posiciones originales. Por ejemplo, el vector

   ```
   {0,11,16,7,1,2,18,14,24,23}
   ```

   queda de la forma

   ```
   {0,11,2,7,1,14,16,18,24,23}
   ```

   Note que los elementos que son pares han sido ordenados de acuerdo al requerimiento de arriba, mientras que los elementos impares no fueron modificados por la función implementada.