Primer Parcial

27 de febrero de 2021

Indicaciones generales

1. Este es un examen **individual** con una duración de **100 minutos: de 10:10am a 11:50am**.
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. **Cualquier incumplimiento de lo anterior conlleva la anulación del examen.**
13. Las respuestas deben estar totalmente justificadas.
14. **Entrega**: archivos con extensión `.cpp` o `.hpp` según el caso, conteniendo el código. Nombre los archivos como `pY.Z`, con `Y = 1,2,3` y `Z = 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. [60 ptos.] Este problema trata de encontrar el elemento que más se repite, es decir, que con más frecuencia aparece, en una lista. El problema puede ser resuelto en dos pasos.

   a) [30 ptos.] Primero, usando `map` construya un histograma de frecuencia de los números contenidos en la lista (un histograma de frecuencia guarda cuántas veces se repite cada número en la lista). Implemente este procedimiento como la función

   ```
   1 map<int, unsigned> freq_histo(list<int> lst);
   ```

   que retorna el mapa que contiene el histograma de frecuencias y `lst` es la lista de entrada. Puede modificar la lista dentro de la función si lo considera necesario.

b) [30 ptos.] Segundo, en este paso se realiza una búsqueda del elemento que más veces se repite en la lista. La estrategia más conveniente es implementar una variación de búsqueda lineal en el histograma de frecuencias. Implemente esta parte como la función

```
1 int most_freq(map<int, unsigned> hist);
```

que retorna el elemento de la lista `lst` con mayor frecuencia (el que más se repite) de acuerdo con el histograma almacenado en el mapa `hist`. El mapa no debe ser modificado dentro de la función.

Por ejemplo, si `lst = {9,4,-6,7,3,2,7,-6,8,7}` el contenido del histograma retornado por la primera función debe ser

```
key   --> 9   4   7   -6   2   3   8
value --> 1   1   3    2   1   1   1
```

y, por lo tanto, el elemento más repetido es `7` (con frecuencia `3`), que es el valor que debe retornar la segunda función.

IMPORTANTE: Note que la implementación de la segunda parte del problema **no depende** de la implementación en la primera parte. Aún si tiene dudas sobre la implementación de la primera parte, implemente la segunda suponiendo que cuenta con el resultado correcto de la primera parte.

2. [40 ptos.] Escriba una función en C++ que recibe un `vector` (`v`) con 3 elementos de tipo `int`. La función debe organizar los 3 elementos en el vector de la siguiente manera: como primer elemento debe ir el elemento menor del arreglo y como segundo elemento debe ir el elemento mayor del arreglo. El prototipo de la función es

```
1 void weird_order(vector<int>& v);
```

Por ejemplo, si el vector contiene `v = {8,5,2}`, la función lo modifica para dejarlo en el orden `v = {2,8,5}`.