Primer Parcial

25 de febrero 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. [30 ptos.] Considere un conjunto de N elementos ordenados de menor a mayor. La mediana del conjunto se define como el elemento medio, cuando N es impar; y como la media de los dos elementos del medio, cuando N es par. Por ejemplo, en la colección de elementos `{-1, 1, 3, 4, 5, 7, 9}` la mediana es 4. Pero para `{7, 8, 9, 10, 11, 12}` la mediana es 9.5.

   Dado un `stack` ordenado de reales (`float`), implemente la función

   ```
   1 float median(stack<float> &S);
   ```

   que recibe como argumento el `stack` y retorna la mediana de los elementos que se encuentran en él. Puede modificar el `stack` si lo considera necesario.

2. [30 ptos.] Dado un `queue` de enteros ordenados, implemente la función

   ```
   1 void invert(queue<int> &Q);
   ```

   que recibe como argumento el `queue` y, *usando un* `stack` de números enteros, invierte el orden de sus elementos. Por ejemplo, si originalmente los elementos en `Q` son `{1, 0, 5, 9, 8}`, luego de invocar `invert` el nuevo contenido de `Q` debe ser `{8, 9, 5, 0, 1}`.

3. [40 ptos.] El sucesor de una llave en un vector de números está definido como el número más pequeño que es estrictamente mayor que la llave.

   Sea `ivec` un *vector ordenado de números enteros no negativos* y `key` una llave entera. Como ejemplo, considere el vector `ivec = {0, 2, 4, 6, 8}`. Si `key = 5` el resultado de la búsqueda debe retornar 3 porque `ivec[3] = 6` es el elemento más pequeño en `ivec` que es mayor que `key`. De manera similar, si ahora `key = 4`, la función debe retornar 3 por la misma razón. Por otro lado, si `key = 9` la búsqueda debe arrojar como resultado -1.

   Usando una modificacion de **búsqueda lineal**, encuentre la posición del sucesor de `key` en `ivec`. Implemente su algoritmo como una función en C++ con el prototipo:

   ```
   1 int lin_succ(const vector<int> &ivec, int key);
   ```

   Si el sucesor existe la función debe retornar su posición en el vector; en caso contrario la función debe retornar -1.