Primer Parcial

31 de agosto 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. [30 ptos.] Para cada cliente de un banco, los consumos con tarjeta de crédito del último mes se almacenan en un `vector<float>`. La tarjeta tiene, además, asociado un cupo total guardado como una variable `cupo_tot`. Con base en lo anterior:

   a) Escriba una función

   ```
   float dispble(vector<float> consumo, float cupo_tot);
   ```

   que reciba el `vector` de consumos y el cupo total, y retorne el cupo disponible; es decir, el cupo total menos el total consumido.

   Ejemplo: Suponga que `consumo = {6.6, 1.1, 4.4, 2.2}`. Si `cupo_tot = 14.0`, la función retorna `14.0 - 14.3 = -0.3` como cupo disponible (la tarjeta de crédito está sobregirada). Por otro lado, si `cupo_tot = 16.8`, retorna como cupo disponible `16.8 - 14.3 = 2.5` (el cliente puede seguir consumiendo).

   b) Escriba otra función

   ```
   int transcc(vector<float> consumo, float cupo_tot);
   ```

   que reciba el `vector` de consumos y el cupo total, y retorne la posición de la última transacción válida que se encuentra por debajo del cupo total. Si el cliente no ha superado dicho cupo entonces la función debe retornar `-1`.

   Ejemplo: Suponga que `consumo = {6.6, 1.1, 4.4, 2.2}`. Si `cupo_tot = 14.0` la función retorna el índice 2 como última transacción válida. Por otro lado, si `cupo_tot = 16.8` todas las transacciones son válidas, entonces se retorna -1.

   El total consumido se puede calcular como la suma de todas las transacciones contenidas en el `vector` de consumos.

2. [30 ptos.] La conjetura de Collatz presenta una manera de generar sucesiones de números naturales que, sin importar con qué número se empiece, siempre termina en 1. Dado un número natural de inicio, la sucesión de Collatz se genera aplicando la siguiente operación a cualquier número natural:

   - Si el número es par, se divide entre 2
   - Si el número es impar, se multiplica por 3 y se le suma 1

   Por ejemplo, si empezamos con $n = 6$ se obtiene la siguiente sucesión de 9 elementos: $\{6, 3, 10, 5, 16, 8, 4, 2, 1\}$. Otros números de inicio que satisfacen la conjetura son $n = 12$ y $n = 19$ con 10 y 21 elementos en la sucesión, respectivamente. Aunque sigue siendo una pregunta abierta si toda sucesión de elementos eventualmente llega a 1, programando este problema es posible dar pistas sobre la validez de la conjetura.

   Para este propósito escriba la función

   ```
   map<int,int> collatz(int n);
   ```

que reciba un entero positivo n y genere una sucesión de Collatz usando a n como número de inicio. La función *crea y retorna* un mapa, en donde los valores son los elementos de la sucesión y las llaves corresponden a la posición de los elementos en la sucesión. En el ejemplo anterior tendríamos el mapa:

```
Keys:   1  2  3  4  5  6  7  8  9
Values: 6  3 10  5 16  8  4  2  1
```

3. [40 ptos.] Considere un `vector<int> vec` de números enteros. Diseñe un algoritmo para encontrar **un** mínimo local en `vec`. Un *mínimo local* de un vector $A = \{A_1, \ldots, A_N\}$ se define como la posición o índice $i \in [0, N-1]$ tal que tanto $A_{i-1} \geqslant A_i$ como $A_i \leqslant A_{i+1}$ son expresiones verdaderas. Implemente su algoritmo como la función

```
int anti_peak(const vector<int> & vec);
```

que toma como parámetro el vector y retorna el índice correspondiente a **un** mínimo local. Note que su algoritmo debe retornar solamente un mínimo local, cualquiera, el primero que encuentre. Note también que pasar por referencia (`&`) y el modificador `const` implican que el vector `vec` no se puede modificar dentro de la función.

Ejemplo: Si $A = \{7, 2, 5, 6, 5, 1\}$, los mínimos locales de A están en las posiciones $i = 1, 5$. Por otro lado, si $A = \{6, 4, 4, 1, 1, 5, 7, 8\}$, los mínimos locales son $i = 1, 3, 4$.