INF-134 Estructuras de Datos, 2016-1

Tarea 3

Profesor: Diego Arroyuelo B.
darroyue@inf.utfsm.cl

Ayudantes:
Francisco Olivares francisco.olivars.14@sansano.usm.cl,
Camilo Saldias camilo.saldias.12@sansano.usm.cl,
Cristian Vallejos cristian.vallejos.14@sansano.usm.cl.

Fecha de entrega: 23 de Junio, 2016
Plazo máximo de entrega: 5 días.

1. Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con los ayudantes ANTES de comenzar la tarea. No se permiten de ninguna manera grupos de más de 3 personas. Las tareas deben compilar en los computadores que se encuentran en el laboratorio LDS – B-032. Deben usarse los lenguajes de programación C o C++. Se recomienda compilar en el terminal usando `gcc archivo.c -o output -Wall` (en el caso de lenguaje C) o `g++ archivo.cpp -o output -Wall` (en el caso de C++). Recordar que a lo más una tarea en el semestre puede tener nota menor a 30. El no cumplimiento de esta regla implica reprobar el curso.

2. Objetivos

Entender y familiarizarse con el uso de heaps y colas de prioridad.

3. Sumando Números

El objetivo de la presente tarea es sumar un conjunto de números dado como entrada. Sin embargo, para que el problema no sea trivial, se agrega un costo a cada operación de suma. Si se suman los números i y j, el costo de la operación es $i + j$. Entonces el problema consiste en sumar el conjunto de números con el menor costo total.

Por ejemplo, si la lista de números es 1, 2 y 3 (lo cual suma 6), entonces tenemos las siguientes alternativas:

- $1 + 2 = 3$, con costo 3, y luego $3 + 3 = 6$ con costo 6. El costo total es $3 + 6 = 9$.
- $1 + 3 = 4$, con costo 4, y luego $4 + 2 = 6$ con costo 6. El costo total es $4 + 6 = 10$.
- $2 + 3 = 5$, con costo 5, y luego $5 + 1 = 6$ con costo 6. El costo total es $5 + 6 = 11$.

Por lo tanto, el costo mínimo para sumar los números 1, 2 y 3 es 9.

Diseñar un algoritmo eficiente que permita decidir en qué orden deben sumarse los números para conseguir el menor costo total. Su algoritmo debe hacer uso de estructuras de dato tipo heaps.

4. Entrada de Datos

La entrada de datos será a través de la entrada standard. Cada caso de prueba comienza con un número entero positivo `N` ($2 \leq$ `N` ≤ 50000), seguido por una línea con los `N` valores enteros positivos que deben sumarse (cada valor es menor o igual a 50000). La entrada es terminada por un caso donde el valor de `N` es cero, el cual no debe ser procesado.

Un ejemplo de entrada válida es:

```
3
1 2 3
4
3 200 1 4
4
1 2 3 4
0
```

Para facilitar la prueba, se recomienda crear un archivo (por ejemplo, `input-edd-t3.txt`), el cual contenga los casos de prueba con el formato pedido. Luego, en la línea de comando (al momento de ejecutar el programa, suponiendo que el ejecutable se llama `tarea`) se redirecciona la entrada standard de la siguiente manera:

```
./tarea < input-edd-t3.txt
```

Aquí, "`<`" indica al sistema que los datos deben ser leídos desde el archivo indicado, y no desde el teclado como es usual. De esta manera, no hay que reingresar los casos de prueba cada vez que se usa el programa.

5. Salida de Datos

La salida de datos se realizará a través de la salida standard. Por cada caso de prueba, se debe imprimir una línea que contenga el costo mínimo de sumar los números correspondientes.

La salida correspondiente a la entrada de ejemplo de la sección anterior es:

```
9
220
19
```

6. Entrega de la Tarea

La entrega de la tarea debe realizarse enviando un archivo comprimido llamado

`tarea3-apellido1-apellido2-apellido3.tar.gz`

(reemplazando sus apellidos según corresponda) en el moodle del curso, a más tardar el día 23 de junio de 2016, a las 23:55:00 hs (Chile Continental), el cual contenga:

- Los archivos con los códigos fuentes necesarios para el funcionamiento de la tarea. Los archivos deben compilar!
- **nombres.txt**, Nombre, ROL, Paralelo y qué programó cada integrante del grupo.
- **README.txt**, Instrucciones de compilación en caso de ser necesarias.

7. Restricciones y Consideraciones

- Por cada día de atraso en la entrega de la tarea se descontarán 10 puntos en la nota.
- El plazo máximo de entrega es 5 días después de la fecha original de entrega.
- Pueden programar la tarea en C o C++ según ustedes consideren conveniente. Al programar en C++ queda prohibido utilizar la librería STL.
- Las tareas deben compilar en los computadores que se encuentran en los laboratorios LDS (B-032). **Las tareas que no compilen no serán revisadas y serán calificadas con nota 0**.
- Por cada *Warning* en la compilación se descontarán 5 puntos.
- Si se detecta **COPIA** la nota automáticamente sera 0 (CERO), para todos los grupos involucrados. El incidente será reportado al jefe de carrera.
- La prolijidad, orden y legibilidad del código fuente es obligatoria. Habrá descuentos si alguno de estos items no se cumple.

8. Consejos de Programación

El código fuente del programa debe estar estructurado adecuadamente en archivos (separados de ser necesario). Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota.

Cada función programada debe tener comentarios de la siguiente forma:

```
/*****
*   TipoFunción NombreFunción
******
*   Resumen Función
******
*   Input:
*       tipoParámetro NombreParámetro : Descripción Parámetro
*        .......
******
*   Returns:
*       TipoRetorno, Descripción retorno
*****/
```

Por cada comentario faltante, se restarán 5 puntos.

Por último, la indentación (1 TAB o 4 espacios), es muy importante. Por **cada bloque mal indentado, se quitarán 10 puntos**.