INF-134 Estructuras de Datos, 2018-1

Tarea 2

Profesores: Diego Arroyuelo, Natalia González, Roberto Díaz
darroyue@inf.utfsm.cl, natalia.gonzalezg@usm.cl, robertodiazurra@gmail.com

Ayudantes:
Camilo Saldias (camilo.saldias.12@sansano.usm.cl)
Abdel Sandoval (abdel.sandoval@sansano.usm.cl)
Ignacio Tampe (ignacio.tampe@sansano.usm.cl)
Alejandro Vilches (alejandro.vilches@sansano.usm.cl).

Fecha de entrega: 3 de agosto 2018
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++).

2. Objetivos

Entender y familiarizarse con la implementación de y uso de estructuras de datos de tipo listas y árboles.

3. Polinomios Usando Listas

A usted y su equipo se les ha encargado programar el software PowerMath, una herramienta matemática similar a WolframAlpha. Para esto tendrá que resolver el siguiente problema haciendo uso de las estructuras lineales vistas en clases.

Un polinomio de grado n corresponde a una expresión matemática de la siguiente forma:

$$a_0x^0 + a_1x^1 + \cdots + a_nx^n,$$

en donde los coeficientes $a_0, a_1, \cdots, a_n$ serán valores enteros. Dada la aplicación en que se usarán los polinomios, no se conoce el grado máximo n que pueden tener, pudiendo ser éste muy grande (aunque se sabe que un `unsigned int` de 32 bits es suficiente para representar n). Haciendo uso de la representación de listas enlazadas, usted deberá representar el TDA polinomio con un conjunto de operaciones adecuado, de manera de poder resolver los requerimientos que se explican más abajo.

Formato de Entrada

Los datos de entrada a su programa serán leídos desde el archivo `entradaPolinomio.txt`, el cual en su primera línea indica un valor entero `N` (menor a 1000), correspondiente a la cantidad de polinomios que se van a leer desde el archivo.

A continuación, en las siguientes líneas del archivo, vienen los datos correspondientes a cada uno de los `N` polinomios. Internamente, cada polinomio deberá ser identificado con un número (comenzando de `0`), correspondiente al orden en que fue leído desde la entrada (el primer polinomio es el `0`, el segundo el `1`, etc.). Los datos de cada uno de los `N` polinomios comienzan con una línea que contiene un valor `M`, que indica cuántos monomios tiene el polinomio. A continuación, las siguientes `M` líneas tienen los datos de los monomios, en el formato `E C`, en donde `E` es el exponente del monomio, y `C` el coeficiente correspondiente. Esos dos valores son separados por un único espacio.

Luego de los datos correspondientes a los polinomios, le siguen una serie de operaciones sobre los polinomios, identificadas como a continuación:

`EVALUAR i X:` produce como resultado la evaluación del polinomio `i` con valor `float` $x =$ `X`. El resultado debe imprimirse con una precisión de hasta 6 digitos decimales.

Para evaluar el polinomio de forma eficiente, debe estudiar e implementar el algoritmo de Horner (ver, por ejemplo, `https://es.wikipedia.org/wiki/Algoritmo_de_Horner`)

`COEFICIENTE i j:` produce como resultado el coeficiente a_j para el polinomio `i`. En caso de no existir el monomio $a_j x^j$ en el polinomio, se produce `0` como resultado.

Un ejemplo de archivo de entrada es el siguiente:

```
3
3
0 3
3 8
1 5
2
2 4
100 -8
2
0 -3
1 3
COEFICIENTE 1 100
EVALUAR 2 3.5
```

Aquí se representan tres polinomios: $3x^0 + 5x^1 + 8x^3$, $4x^2 - 8x^{100}$, y $-3x^0 + 3x$, respectivamente.

El archivo de entrada es terminado por el fin de archivo `EOF`.

Formato de Salida

Por cada operación especificada en el archivo de entrada, debe escribirse el resultado en el archivo `salidaPolinomio.txt`. El resultado de una operación se imprime en una línea separada.

Para el archivo de entrada presentado anteriormente, la salida correspondiente debe ser:

```
-8
7.5
```

4. Polinomios Usando Árboles Binarios de Búsqueda

En esta sección se deben soportar las mismas operaciones que en la Sección 3, pero esta vez implementando polinomios usando árboles binarios de búsqueda. Se debe diseñar la manera en que un polinomio será almacenado en la estructura de datos, para soportar las operaciones de manera eficiente.

5. Entrega de la Tarea

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

`tarea2-apellido1-apellido2-apellido3.tar.gz`

(reemplazando sus apellidos según corresponda) en el sitio Aulas USM del curso, a más tardar el día 3 de agosto 2018, 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.

6. 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 el laboratorio 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.

7. 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**.