INF-134 Estructuras de Datos, 2018-2

Tarea 2

Profesora: Natalia González.
natalia.gonzalezg@usm.cl

Ayudantes:
Camilo Saldías (camilo.saldias.12@sansano.usm.cl)
Alejandro Vilches (alejandro.vilches@sansano.usm.cl)

Fecha de entrega: 18 de noviembre, 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. No se permiten de ninguna manera grupos de más de 3 personas. Las tareas deben compilar en los computadores que se encuentran en el laboratorios LDS – B-032 (Fedora 17 de 64 bits). Debe usarse el lenguaje 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 una única 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 estructuras de datos lineales, en este caso listas, pilas y colas. Además se fomentará el uso de las buenas prácticas y el orden en la programación de los problemas correspondientes.

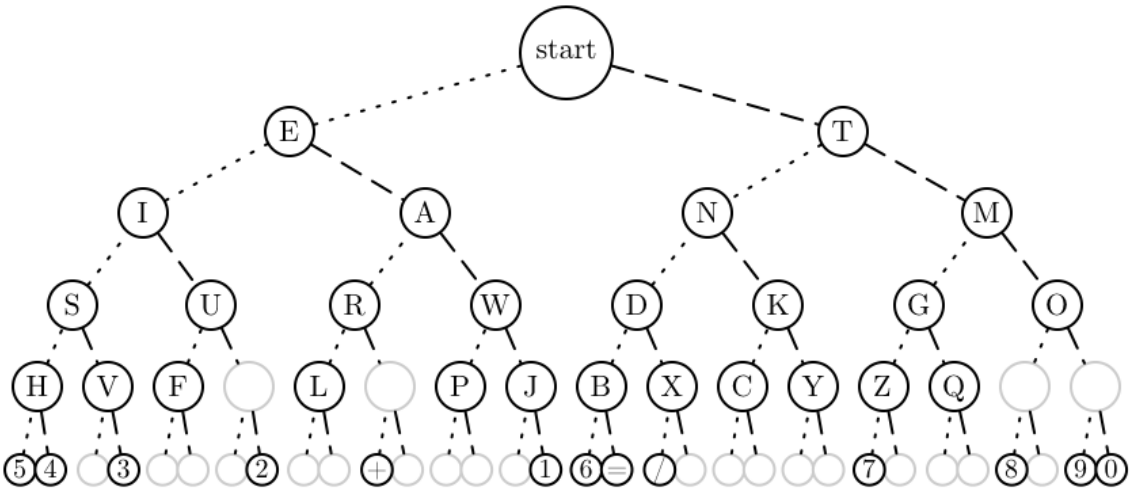
3. Problemas a Resolver

Problema: Código Morse

El código Morse fue desarrollado por Alfred Vail en 1830, mientras colaboraba con Samuel Morse en la invención del telégrafo eléctrico. Consiste en un método según el cual cada letra o número era transmitido de forma individual con un código consistente en rayas y puntos, las cuales eran fácilmente transmisibles e identificables al ser usadas en líneas telegráficas.

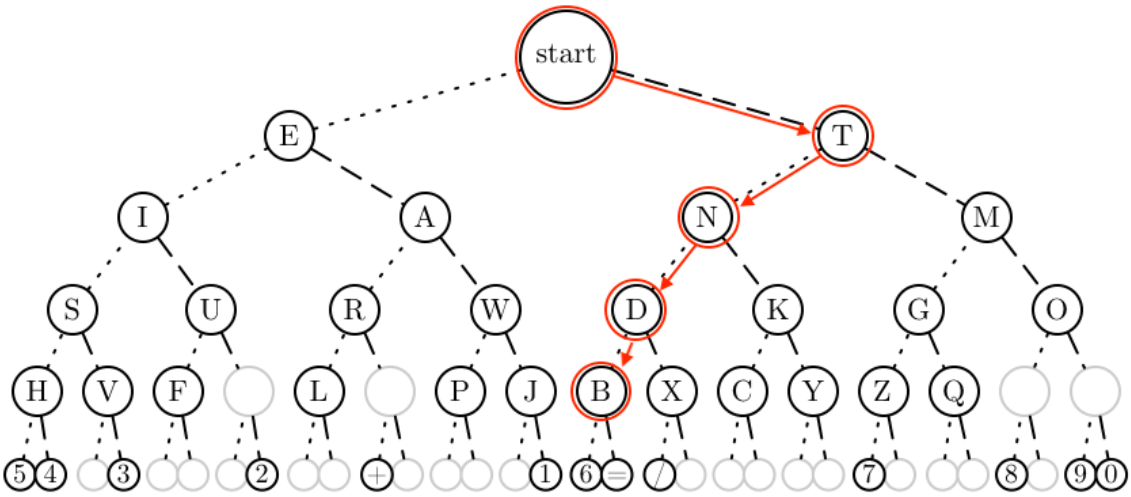
Las asociaciones entre los caracteres del alfabeto norteamericano y su equivalencia en código Morse son bien conocidas, y pueden ser listadas en forma de tabla para su consulta. Sin embargo, esto imposibilita la rápida decodificación de los mensajes recibidos en código Morse, dado que se debe esperar a la recepción de cada caracter para su posterior decodificación.

Es por lo anterior que se propuso utilizar el siguiente árbol, el cual permite la decodificación de caracteres a medida que son recibidos en código Morse:



Para decodificar un caracter codificado en código Morse, basta con comenzar desde el nodo raíz y desplazarse a través del árbol siguiendo la secuencia del código Morse, tomando el camino derecho si el caracter es un punto y el camino izquierdo si el caracter es una línea. Al finalizar el recorrido, el nodo final representa la letra correspondiente a esa codificación Morse.

A continuación se presenta un ejemplo de lo anterior, utilizando la letra B (-...) como ejemplo a decodificar:



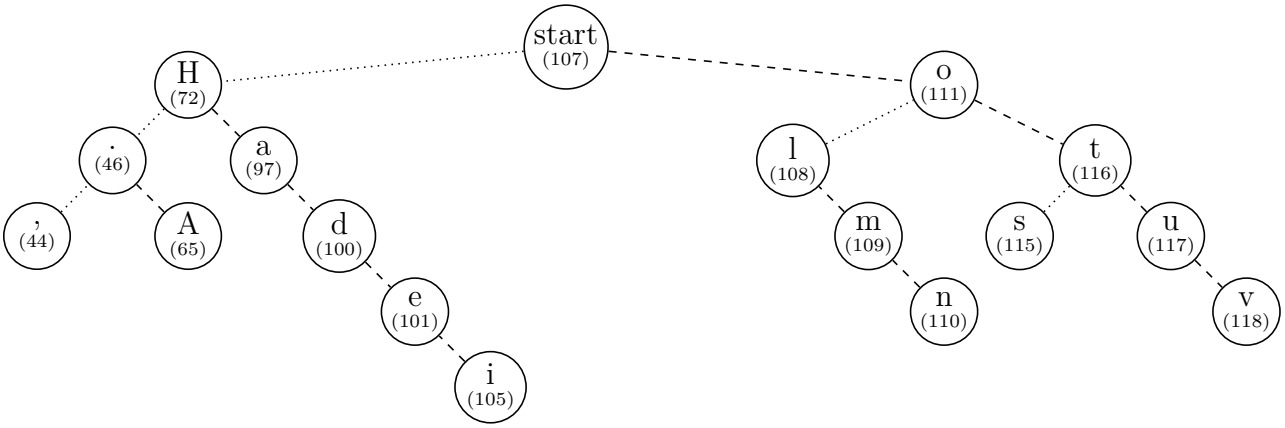
En esta ocasión usted deberá crear su propia codificación para mensajes, creando el programa `codificador`. La diferencia será que el árbol sólo estará formado por los caracteres presentes en el mensaje a codificar. El orden de las letras estará dado por el valor del código ascii correspondientea cada caracter, e ingresarán a un ABB (respetando el orden usual de ABB) considerando como el orden de ingreso al orden en que aparecen los caractéres en el mensaje. Al nodo raíz (start) se le asignará por defecto la letra “k” (valor: 107).
Por ejemplo para codificar el mensaje:

Hola a todo el mundo. Adios, nos vemos.

tenemos que los caracteres (con su correspondiente valor ascii) y respetando el orden de aparición son:

(*i*) H (72)	(*v*) t (116)	(*vii*) m (109)	(*xii*) A (65)	(*xvi*) v (118)
(*ii*) o (111)	(*vi*) d (100)	(*ix*) u (117)	(*xiii*) i (105)	
(*iii*) l (108)		(*x*) n (110)	(*xiv*) s (115)	
(*iv*) a (97)	(*vii*) e (101)	(*xi*) . (46)	(*xv*) , (44)	

Por lo que el ABB correspondiente debería quedar:



Donde, por ejemplo, *"Hola"* corresponderá a "`. - -. .-`"

Formato de Entrada

La entrada de datos será a través de un archivo llamado "`mensaje.txt`" con un mensaje a codificar, siendo el archivo terminado por `EOF`. El mensaje a codificar es una única línea que contiene al menos un y a lo más 100.000 caracteres. Los caracteres permitidos son las letras del abecedario (a excepción de los caracteres "k", "ñ" y "Ñ"), todos los dígitos, el punto (" . ") y la coma (" , "). No estará permitido el uso de tildes (u otros signos de acentuación).
El nombre de este archivo será ingresado como argumento al momento de ejecutar su programa, de la siguiente forma:

```
./codificador "mensaje.txt"
```

Para leer el nombre del archivo pasado como argumento, usted debe poder aceptar parámetros en la función `main` de su programa, lo que se realiza de la siguiente forma:

```
int main ( int argc, char *argv[] )
```

Donde `argc` es la cantidad de argumentos entregados, y el arreglo `argv` contiene los argumentos entregados. Por defecto, el primer argumento (en la posición 0 del arreglo `argv`) es el nombre del programa, por lo que el valor por defecto de `argc` es 1.

Notar que en el caso de que el archivo indicado como argumento no exista, el programa debe indicar por pantalla '`Archivo no existe`' y terminar su ejecución.

Formato de Salida

El programa debe imprimir en el archivo "salida.txt" el mensaje codificado, donde con el caracter "`/`" se deben separar las palabras del mensaje. La salida correspondiente al ejemplo anterior (*Hola a todo el mundo. Adios, nos vemos.*) es la siguiente:

```
. - -. .-/.-/-- - .-- -/.--- -./-.- --- -.-- .-- -/../..- .-- .---- - --./
.../-.-- - --./---- .--- -.- - --./../
```

4. 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) al sitio de AULA del curso, a más tardar el día 18 de noviembre de 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, y qué programó cada integrante del grupo.
- **README.txt**, Instrucciones de compilación en caso de ser necesarias.

4.1. Entrega Temprana

Se dará la opción de realizar entrega de la versión final de la tarea de manera adelantada, bonificando su nota de la siguiente manera:

1. Entregas hasta el 11/11 a las 23:55:00 hrs via Aula: Bonificación de +10 pts en la nota de la tarea.
2. Entregas hasta el 14/11 a las 23:55:00 hrs via Aula: Bonificación de +5 pts en la nota de la tarea.

5. 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.
- Las tareas deben compilar en los computadores que se encuentran en los laboratorios LDS (Fedora 17 de 64 bits). **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 será 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.

6. 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**.