Department of Computing & Information Systems
Trent University

COIS 2020H: Data Structures and Algorithms

Assignment 3: File Systems

Due date: Monday, December 5, 2016 at midnight
No assignment can be accepted after midnight on December 5, 2016

The common computer file system is similar to a tree where each node represents a directory that contains 0 or more files, and 0 or more links to subdirectories.

To implement such a file system, a data structure called the **leftmost-child, right-sibling** is often used. The leftmost child refers to the first subdirectory of a node, and the rightmost child refers to the next directory on the same level as the node (see Figure 1). In this case, the root directory has three files (P, Q and R) and three subdirectories (A, B and C). Note that directory names can be repeated as long as they do not appear as siblings. Similarly, file names can be repeated as long as they do not appear in the same directory.

The primary task of this assignment is to design, implement, and test a file system using the leftmost-child, right-sibling as the data structure and where the behaviour of the file system is defined by the methods in Figure 2 (next page). Each method in FileSystem is worth 7 marks, except the constructor which is worth 3 marks.

Figure 1

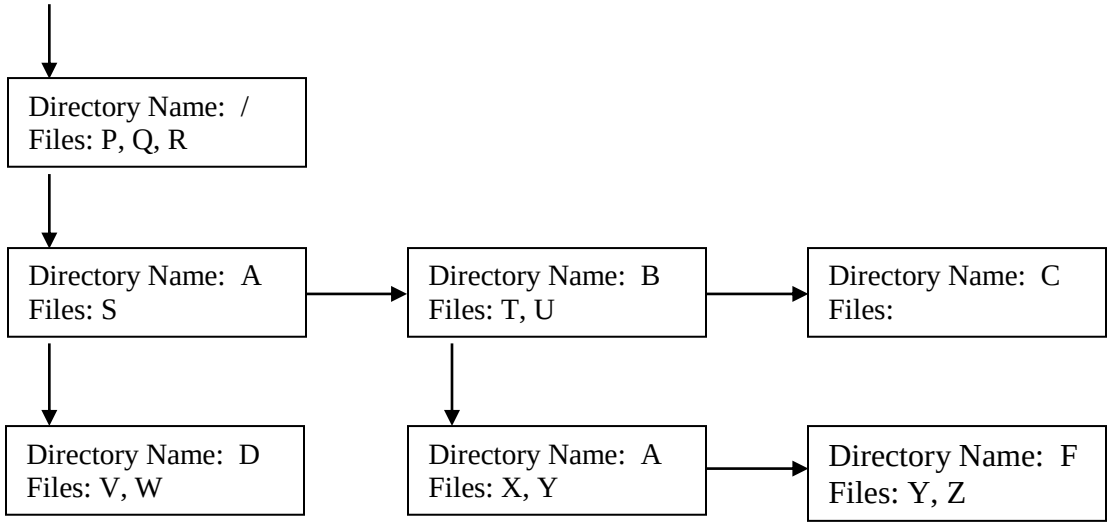

Figure 2

```
class Node
{
     private string directory;
     private List<string> file;
     private Node leftMostChild;
     private Node rightSibling;
     ...
}

public class FileSystem
{
     private Node root;

     // Creates a file system with a root directory
     public FileSystem( ) { ... }

     // Adds a file at the given address
     // Returns false if the file already exists or the path is undefined; true otherwise
     public bool AddFile(string address) { ... }

     // Removes the file at the given address
     // Returns false if the file is not found or the path is undefined; true otherwise
     public bool RemoveFile(string address) { ... }

     // Adds a directory at the given address
     // Returns false if the directory already exists or the path is undefined; true otherwise
     public bool AddDirectory(string address) { ... }

     // Removes the directory (and its subdirectories) at the given address
     // Returns false if the directory is not found or the path is undefined; true otherwise
     public bool RemoveDirectory(string address) { ... }

     // Returns the number of files in the file system
     public int NumberFiles( ) { ... }

     // Prints the directories in a pre-order fashion along with their files
     public void PrintFileSystem( ) { ... }

}
```

For each of the methods above, the address for a directory or file is given in absolute terms (i.e. given in full from the root). For instance, the address of a directory or file is stated as:

```
/ADirectory/BDirectory/CDirectory
/ADirectory/BDirectory/CDirectory/AFile
```

Your program will need to parse the given address to determine which path to follow in the file system.

What to Submit

As usual, submit the source and executable files as well as the test results both digitally **and** in hard copy.

Grading Scheme

File class	5%
Parsing the address	10%
Methods of FileSystem	45%
Main (Interface)	15%
Testing	15%
Inline documentation	10%