CS 3345 Data Structures Project 1

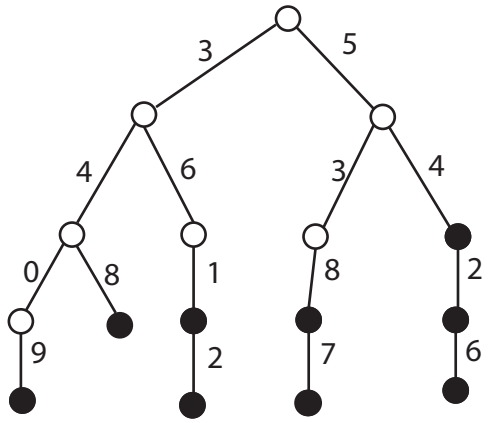
You are going to write a class to implement a set for integers. The *Trie data structure* will be used to implement the set. Your program will include a main section that reads commands from `System.in`. See the sample at the end of this spec'.

The Trie will store numeric (integer) keys of up to 30 digits.

The member functions of the Trie class will be:

```
boolean insert(String s);      // String s comprises a decimal integer of up to 30 characters.
                               // insert returns false if s is already present, true otherwise
boolean isPresent(String s);   // returns true if s is present, false otherwise
boolean delete(String s);      // returns true if s was present, false otherwise
int membership();              // returns the number of keys in the data structure
void listAll();                // list all members of the Trie's tree in depth first search order
```

A Trie is a tree of degree 10. The following Trie contains the keys, listed here in DFS order: 3409, 348, 361, 3612, 538, 5387, 54, 542, 5426.



The black Nodes have a boolean variable set to indicate the end of a key value.

Here are the class variables for the class Node:

```
class Node {
   boolean terminal;
   int outDegree;
   children Node[];
}
```

Each Node includes an array of 10 references to Node. When a Node is created, all the array elements in that Node will be initialized to null and the outDegree will be set to zero. You may add other variables to the Node class, but do not add a reference to the Node's parent and do not store a key value in a Node in any form.

In the above Trie, the root node has only two children, corresponding to digit values '3' and '5', or positions 3 and 5 in the array of references.

A search begins at the root Node and follows the links, one level for each digit in the given key. To find the key “348” we begin with array element 3 in the root Node and, if it is not null, we check element 4 in the array of the second level node. Finally, we check element 8 in the array of the third level Node. If that Node is marked as a key-terminal, we return “true”.

The insert function begins with a search for the given key. If a null reference is found, a sequence of new Nodes is created corresponding to the missing digits of the key ending.

To insert the key “367” to the above Trie, one new node would be added, corresponding to ’7’. This new node would be referenced by the third level Node corresponding to the prefix “36”.

Deletion also begins with a search for the given key. If found, there are two cases. Say we are deleting “361” from the above Trie. Since the terminal character of the given key has children, deletion only requires flipping the boolean variable that indicates the end of a key.

If the key “348” is deleted, the suffix “8” must be unlinked from the third level node. To do this, when recursing out from the end of the given key, remove links to the characters of that key until a Node is found with degree greater than 1, or a terminal node is found.

Your program will read a text file from System.in that contains text commands. In response to each command your program will output one or more lines of text to system.out.

Here are the commands:

```
N         // Print your name followed by a newline.
A 1234    // Insert the key ’1234’
          // Print one of the strings "key 1234 inserted" or "key 1234 already exists"
          // followed by a newline.
D 344     // Delete the key ’344’.
          // Print "key 344 deleted" or "key 344 not found" followed by a newline.
S 98765   // Search for the key "98765".
          // Print "key 98765 found" or "key 98765 not found" followed by a newline.
M         // Print "Membership is N" where N is the number of keys in the Trie
C  ka kb kc kd ke  // Check the space separated sequence of keys ka, kb, kc, kd, ke, ...
          // up to the end of the line for presence in the Trie and only list those that are
          // not present, one per line, each in format "Key kx not found".
          // The keys in the list may each be up to 30 characters long and the input line may
          // be up to 200 characters long.
L         // Print a list of all the elements of the Trie in depth first search order, one key per line.
E         // The end of the input file
```

Here are sample input and output data files:

Sample Input	Output for Sample Input
N	Ivor Page
A 123	Key 123 inserted
A 2143	Key 2143 inserted
A 4532	Key 4532 inserted
A 123	Key 123 already exists
A 2415	Key 2415 inserted
A 26145	Key 26145 inserted
A 12	Key 12 inserted
A 38	Key 38 inserted
A 24	Key 24 inserted
A 45	Key 45 inserted
A 38	Key 38 already exists
A 453	Key 453 inserted
A 26	Key 26 inserted
A 261457	Key 261457 inserted
A 261	Key 261 inserted
A 2614	Key 2614 inserted

```
A 7755                                  Key 7755 inserted
A 145                                   Key 145 inserted
M                                       Membership is 16
D 26                                    Key 26 deleted
D 26145                                 Key 26145 deleted
D 261457                                Key 261457 deleted
D 145                                   Key 145 deleted
D 2415                                  Key 2415 deleted
S 241                                   Key 241 not found
S 38                                    Key 38 found
S 2415                                  Key 2415 not found
D 145                                   Key 145 not found
S 6194                                  Key 6194 not found
S 85588537129                           Key 85588537129 not found
M                                       Membership is 11
S 6194                                  Key 6194 not found
S 6174                                  Key 6174 not found
S 7745                                  Key 7745 not found
S 67677                                 Key 67677 not found
C 38 145 8124 8888                      Key 145 not found
C 9124 145 85588537129 1111 2222        Key 8124 not found
L                                       Key 8888 not found
E                                       Key 9124 not found
                                        Key 145 not found
                                        Key 85588537129 not found
                                        Key 1111 not found
                                        Key 2222 not found
                                        12
                                        123
                                        2143
                                        24
                                        261
                                        2614
                                        38
                                        45
                                        453
                                        4532
                                        7755
```

RULES FOR PROGRAMMING AND SUBMISSION:

1. Your submitted code must be entirely your own, except for any code that you copy from this specification.
2. Write your program as ONE source file and do not use the "package" construct in your Java source.
3. Name your source file as $N_1N_2F_1F_2$P1.java where your given name begins with the characters N_1N_2 and your family name begins with the characters F_1F_2. For example my name is Ivor Page, so my source file will be called IVPAP1.java. Note that in all but the "java" extension, all characters are upper case.
4. Your program must not output any prompts. Its output must exactly match the format of the **Sample Output** above.
5. Do not use any Java Collection Classes, except the arrays to store references in each node. Strings are allowed for input of command lines.
6. YOUR PROGRAM MUST READ FROM System.in AND OUTPUT TO System.out. See the example below
7. Use good style and layout and comment your code well.

8. Use the test files provided on the eLearning webpage for this class to test your program. Sample input and corresponding output files will be given.

9. Submit your ONE source code file to the eLearning Assignment Dropbox for this project. Don't submit a compressed file. Don't submit a .class file.

10. **There will be a 1% penalty for each minute of lateness, up to 60 minutes. After 60 minutes of lateness a grade of zero will be recorded.**

Sample Sketch of main section
You don't have to follow this exactly. You might read entire lines and use the String.split() function.

```
public static void main( String [ ] args )
{
    Trie tre = new Trie();

    Scanner in = new Scanner(System.in);
    boolean notDone = true;

    while(notDone) {                    // read commands and obey them
        String command = in.next();
        char com = command.charAt(0);
        switch(com) {
        case 'N': {
            System.out.println("Ivor Page");
            break;
        }
        case 'A': {                 // insert
            String toAdd = in.next();
            if(tre.insert(toAdd)) {
                System.out.println("Key " + toAdd + " inserted");
            }
            else
                System.out.println("Key " + toAdd + " already exists");
            break;
        }
        case 'D': {                 // delete
        .  .  .
```