ITU Computer Engineering Department

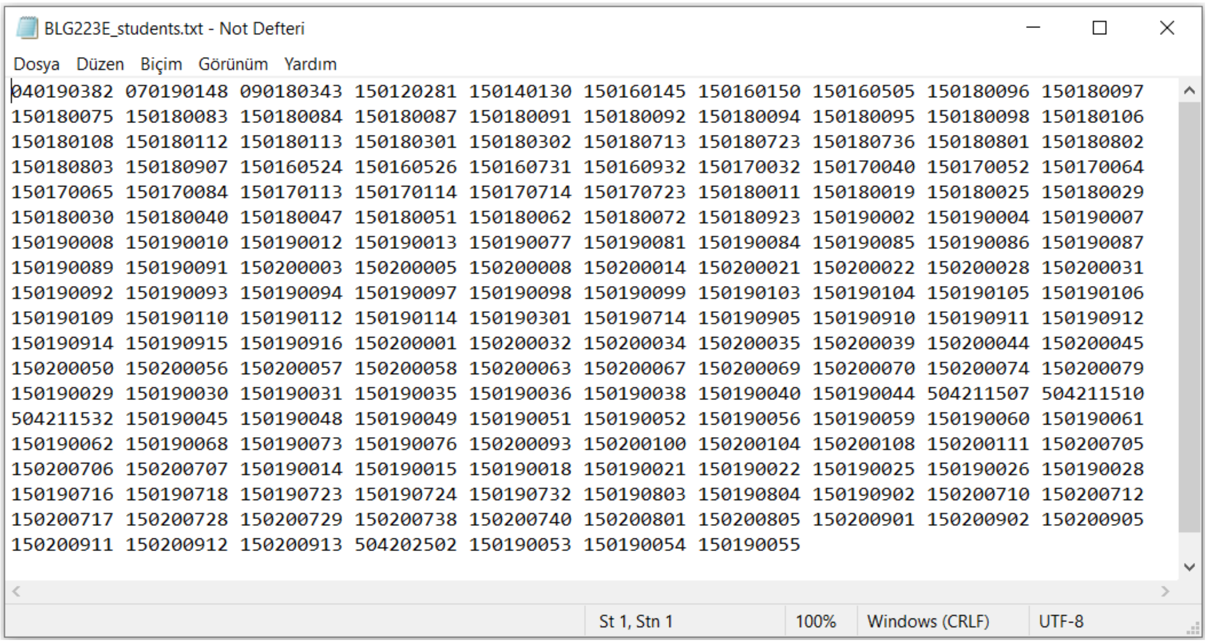
BLG 223E Data Structures, Fall 2021-2022

Recitation #5

Due Date: 29.12.2021, 23:59

You are asked to construct a trie to keep the list of BLG223E students using their student numbers. The trie will be used to search for an enrolled student with his/her student number.

Input: The list of student numbers that enrolled in the BLG223E class are stored in a text file whose name will be given as a **command line argument** (e.g., BLG223E_students.txt). The file contains exactly 10 student numbers per line (except the last line that contains at least 1 and at most 10 student numbers) which are separated by a space character, as shown in Figure 1. All the student numbers consist of 9 digits.


BLG223E_students.txt - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
St 1, Stn 1
100%
Windows (CRLF)
UTF-8

```
040190382 070190148 090180343 150120281 150140130 150160145 150160150 150160505 150180096 150180097
150180075 150180083 150180084 150180087 150180091 150180092 150180094 150180095 150180098 150180106
150180108 150180112 150180113 150180301 150180302 150180713 150180723 150180736 150180801 150180802
150180803 150180907 150160524 150160526 150160731 150160932 150170032 150170040 150170052 150170064
150170065 150170084 150170113 150170114 150170714 150170723 150180011 150180019 150180025 150180029
150180030 150180040 150180047 150180051 150180062 150180072 150180923 150190002 150190004 150190007
150190008 150190010 150190012 150190013 150190077 150190081 150190084 150190085 150190086 150190087
150190089 150190091 150200003 150200005 150200008 150200014 150200021 150200022 150200028 150200031
150190092 150190093 150190094 150190097 150190098 150190099 150190103 150190104 150190105 150190106
150190109 150190110 150190112 150190114 150190301 150190714 150190905 150190910 150190911 150190912
150190914 150190915 150190916 150200001 150200032 150200034 150200035 150200039 150200044 150200045
150200050 150200056 150200057 150200058 150200063 150200067 150200069 150200070 150200074 150200079
150190029 150190030 150190031 150190035 150190036 150190038 150190040 150190044 504211507 504211510
504211532 150190045 150190048 150190049 150190051 150190052 150190056 150190059 150190060 150190061
150190062 150190068 150190073 150190076 150200093 150200100 150200104 150200108 150200111 150200705
150200706 150200707 150190014 150190015 150190018 150190021 150190022 150190025 150190026 150190028
150190716 150190718 150190723 150190724 150190732 150190803 150190804 150190902 150200710 150200712
150200717 150200728 150200729 150200738 150200740 150200801 150200805 150200901 150200902 150200905
150200911 150200912 150200913 504202502 150190053 150190054 150190055
```

Figure 1: Student numbers of the students are kept in a text file

Based on our example BLG223E_students.txt file, Figure 2 shows the partly constructed trie when the student numbers in the first line of the BLG223E_students.txt file are inserted into the trie. When the trie construction is fully completed, the trie will contain nodes for all the student numbers in the BLG223E_students.txt file.

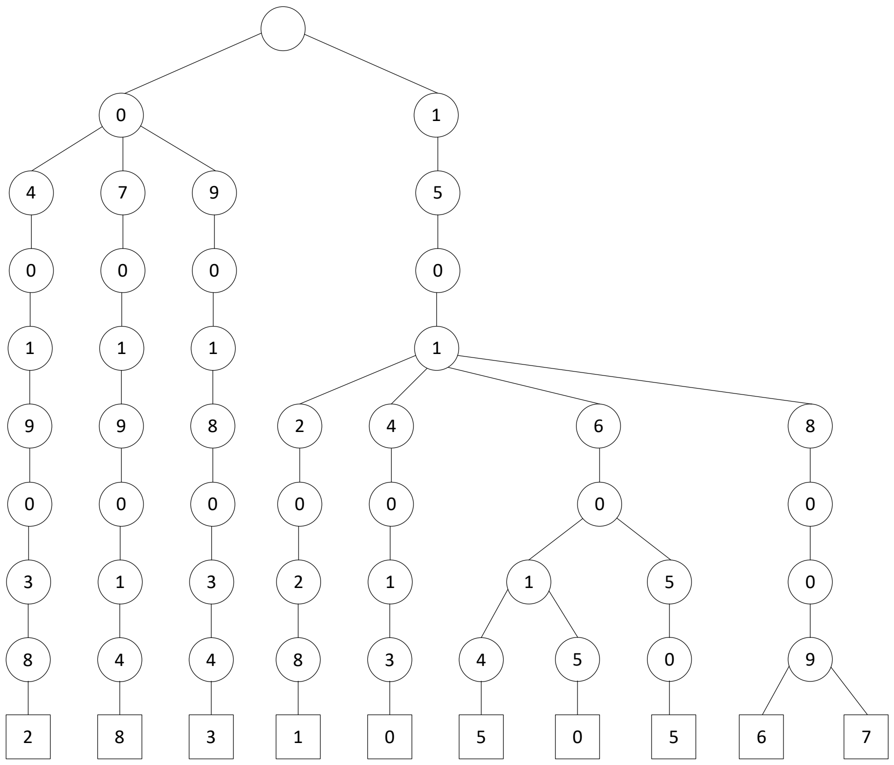


Figure 2: Trie after processing numbers in the first line of example BLG223E_students.txt text file

Wanted: You are given the following C++ StudentsTrie.h header file that defines StudentsTrie data structure along with TrieNode (Figure 3).

```
// PLEASE DO NOT REMOVE DECLARED VARIBLES AND FUNCTIONS OF CLASSES,
// IF YOU PREFER, YOU CAN ADD NEW ONES
#ifndef _H
#define _H

#include <string>
using namespace std;

#define MAX_CHILDREN 10 // Each student ID consists of a sequence of digits: 0-9

class TrieNode{
        char digit;   // Current Digit
        TrieNode* children[MAX_CHILDREN];   // Next Digit(s)
        friend class StudentsTrie;

    public:
        TrieNode(const char& digit);
};

class StudentsTrie{
        TrieNode *root;

    public:
        StudentsTrie     ( const string& file_name  ); // Construct a StudentsTrie using the records in 'file_name'
        void insert_id   ( const string& student_id ); // Insert a student ID into the StudentsTrie
        void print_trie  (                          ); // Print Student IDs in the StudentsTrie in ascending order
        ~StudentsTrie    (                          ); // StudentsTrie Destructor
};

#endif
```

Figure 3: StudentsTrie.h Header File for StudentsTrie data structure

You must implement the methods given in the StudentsTrie.h header file in a StudentsTrie.cpp file so that the following main.cpp file (Figure 4) works as described below.

```
1   /*
2   PLEASE DO NOT CHANGE THIS FILE, OTHERWISE YOUR CODE WILL NOT BE EVALUATED AND WILL BE GRADED AS 0
3   */
4
5   #include <iostream>
6
7   #include "StudentsTrie.h"
8
9   using namespace std;
10
11  int main(int argc, char* argv[]){
12      //system("clear");// make this line as comment if you are compiling on Windows
13      //system("cls"); // make this line as comment if you are compiling on Linux or Mac
14
15      StudentsTrie st(argv[1]);
16
17      st.print_trie();
18
19      return EXIT_SUCCESS;
20  }
```
Figure 4: main.cpp File

The pseudocode for the StudentsTrie constructor (Line 15) is given below:

```
(Step 1)  Get the file_name as an actual parameter (in our case, argv[1]).
(Step 2)  Open the file_name in read mode.
(Step 3)  Read a student_number from the file.
(Step 4)  Insert student_number to the trie using insert_id() method of the StudentsTrie class
          where student_number is the last student number read in (Step 3).
(Step 5)  If there are more student numbers in the file_name, continue with (Step 3) to read next
          student_number. Otherwise, continue with (Step 6).
(Step 6)  Close the file_name and finish constructor processing.
```

After constructing the StudentsTrie data structure, you must traverse the trie and print the student numbers added to trie in ascending order. In order to do it, you have to implement **print_trie()** method of StudentsTrie class (**Line 17**).