Aim: In this assignment, given some patterns and a long text, we ask to implement the Aho-Corasick algorithm and search for the patterns in the text. You can assume that the patterns and text are given properly. The index starts from 0 and space will be considered as a character. In the input file, patterns that are separated by space are given in the first line and the text is given in the second line.
In the build tree part of output, each row will contain **char** that is the character the node represents, **next states** that is a list of the ids of the child nodes, **fail state** that is the id of the fail state, and **output** that is a list of all the complete keywords encountered so far as we have gone through the input text. In the search part of the same output file, each row will contain a keyword and its index. An example is given below:

Input format:

```
carrot cool parrot car carpet rotate
once upon a time there was a cool parrot who loves eating carrot on the carpet
```

Output format:

```
Build tree
---------------------------------------------------------------------
char:  next states: [1, 10, 19] fail state: 0 output: []
char: c next states: [2, 7] fail state: 0 output: []
char: a next states: [3] fail state: 0 output: []
char: r next states: [4, 16] fail state: 19 output: ['car']
char: r next states: [5] fail state: 19 output: []
char: o next states: [6] fail state: 20 output: []
char: t next states: [] fail state: 21 output: ['carrot']
char: o next states: [8] fail state: 0 output: []
char: o next states: [9] fail state: 0 output: []
char: l next states: [] fail state: 0 output: ['cool']
char: p next states: [11] fail state: 0 output: []
char: a next states: [12] fail state: 0 output: []
char: r next states: [13] fail state: 19 output: []
char: r next states: [14] fail state: 19 output: []
char: o next states: [15] fail state: 20 output: []
char: t next states: [] fail state: 21 output: ['parrot']
char: p next states: [17] fail state: 10 output: []
char: e next states: [18] fail state: 0 output: []
char: t next states: [] fail state: 0 output: ['carpet']
char: r next states: [20] fail state: 0 output: []
char: o next states: [21] fail state: 0 output: []
char: t next states: [22] fail state: 0 output: []
char: a next states: [23] fail state: 0 output: []
char: t next states: [24] fail state: 0 output: []
char: e next states: [] fail state: 0 output: ['rotate']

Search
---------------------------------------------------------------------
keyword: cool index: 29
keyword: parrot index: 34
keyword: car index: 58
keyword: carrot index: 58
keyword: car index: 72
keyword: carpet index: 72
```

Command format:

- gcc ${c_file(s)}.c -o ${executable_file_name}

- g++ ${cpp_file(s)}.cpp -o ${executable_file_name}

Notes:

- Your code must be written by yourself. Sufficient evidence of plagiarism will be treated the same as for plagiarism or cheating.
- **Non-compiling submissions will not be evaluated.**
- Your code must be complete.
- **Do not submit the program binary.**
- You must submit the following items:
  (1) **All of the header and source files** (.h & .cpp) or (.h & .c)
  (2) **A script to compile the source code and produce the binary (Makefile**).
  If you do not know how to create makefile, you can use this simple guide or google it:
  http://mrbook.org/blog/tutorials/make/
- Submit your answers to Fatma Kahveci at ' fatma.balci@bilkent.edu.tr '.
- Use ' **CS481 Assignment-2** ' in the subject line of your e-mail.
- Zip your files and send them **in only one zipped file**.
  File name format='**surname_name_hw2.zip**'
- Name your source and header files **properly**.
- **C / C++**, python, or java will be used as programming language.
- **Do not add input and output files.** You can assume that input file is found in the same directory with your header and sources files.
- All submissions must be made by **23:59**, **October 26, 2017**.
- **Bonus** will be given for the fastest code. The fastest wins.