1. **Generalized suffix Tree**

- Implement a suffix tree data structure for a given string to perform string matching for given input text and pattern.
- You may use a quadratic algo in time and space.
- You must build the suffix tree for the whole document given(**NOT for each word in the document**)
- **Input format:**
  The **first line specified is the input text** on which string matching should be performed. The max length of the input text will be 4000 and will not contain **$**.
  M is a valid integer <=100.
  The next M lines are strings of max length 1000 which are patterns to be matched against the input text given.

- **Output format:**
  There will be M lines, each line printing the list of index of the location of the **caseless(case insensitive)** match of the pattern(**which is not limited to being a word and can be a phrase**) with the input text, a space in between them and in increasing order(0 indexed). If the pattern doesn't match anything print -1

- **Sample Input file:**
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis nec placerat augue, sit amet pharetra enim. In ac nulla ut est commodo tincidunt vel nec dolor. Maecenas dui tellus, iaculis eget rutrum suscipit, aliquet congue nisl. Pellentesque eleifend, magna eu pretium sagittis, tortor lacus luctus lorem, a cursus turpis nisl id massa. Nullam eget orci id nisl tincidunt vestibulum tincidunt eleifend mi. Integer justo ipsum, pellentesque vel commodo a, sagittis non augue. Curabitur vulputate libero lorem, sed congue nisi ultricies sit amet. Pellentesque rutrum pellentesque congue
  3
  Lorem
  Qwerty
  Lacus

- **Sample Output file:**
  0 300 503
  -1
  287

2. **String Matching:**

Write a program to find **all** the occurrences of the given pattern in a the input text using the **KMP** algorithm.

Input format:
First line contains the input text which will have a max length of 4000 characters
Second Line contains the number of test cases T (1<= T <= 10^6).
For each test case, there is a pattern to be matched against the input text. The pattern will have a max length of 1000 characters.

Output Format:
Return all indexes of the pattern found in the text for each test case.(0 indexed)
If the text doesn't contain the pattern, return -1.

Sample input:
bananabano
3
na
boo
bano

Sample output:
2 4
-1
8

EDIT 1 *Oct 23*: **Specified the max lengths of pattern and text in the second question**
EDIT 2 *Oct 25*: **Specified that**

- **suffix tree must be built on the document as a whole**

- **Matching must be case insensitive(Added the word case insensitive)**
- **Pattern could be a phrase and is not limited to being a word**