Code Explanation

In my code implementation, it facilitates to perform following regex patterns.

- Plain text: Matches any word without special regex characters
- . (dot): Matches any single character except a newline.
- *: Matches zero or more occurrences of the preceding character or group.
- +: Matches one or more occurrences of the preceding character or group.
- ?: Matches zero or one occurrence of the preceding character or group.

At a time, you can use only one special regex character.

For example, cre*k is valid.
c.e*k is not valid within this program, as it contains both '.' and '*' in the same pattern.

For implementing these regex patterns, I used **Knuth Moris Pratte(KMP)** algorithm.

Why Knuth Moris Pratte(KMP) algorithm?

- ✓ KMP is efficient, with a time complexity of O(n + m).
- ✓ KMP minimizes backtracking, which can lead to inefficient execution.
- ✓ KMP is deterministic, meaning that its behavior is solely determined by the pattern itself.
- ✓ It is memory efficient, compared to backtracking-based algorithms.
- ✓ KMP is well-suited for simple regular expressions.
- ✓ KMP has predictable performance.

Overall, KMP is a good choice for efficiently searching simple regular expression patterns within text. It is efficient, minimizes backtracking, and is deterministic. It is also memory efficient and has predictable performance.

In my code I implemented several functions to perform different regex patterns. Here are the functions:

```
void createLpsArray(char *pattern, int *lps, int patLen);
void KMPSearch(char* pattern, char* txt, int lineNumber);
void KMPSearch_dot(char* pattern, char* txt, int dotIndex, int lineNumber);
void KMPSearch_astrix(char* pattern, char* txt,char precedigCharacter, char
postAstrixCharacter, int astrixIndex, int lineNumber);
void normalSearch(string pattern, char *stringLine, int lineNum);
void dotSearch(string pattern, char *stringLine, int lineNum);
void astrixSearch(string pattern, char *stringLine, int lineNum);
void plusSearch(string pattern, char *stringLine, int lineNum);
void questionSearch(string pattern, char *stringLine, int lineNum);
void appendToFile(int lineNumber, int index);
```

How these functions perform?

First, we need to give input text file and regex pattern. Then the program find is there any special regex character in given input pattern.
Then program read line by line our input text file to detect patterns in our file. Here is the process for one line to detect patterns.

If there is no special regex pattern character in given pattern,

Main function calls for normalSearch function by giving pattern, string line and line number in our text file.
Then normalSearch calls for KMPSearch function by passing pattern, string line and line number. Then KMP function call for createLpsArray function to make longest prefix suffix table. With help of createLpsArray function, KMPSearch function generates the lps array and perform on given string line to find the index of the words where the given string patterns are existed. After finding the places of string patterns KMPSearch calls for appendToFile function. Then appendToFile function write that result into a output.txt file.

If there exist a '.' in given pattern,

It means our regex program Matches any single character except a newline.
Main function calls for dotSearch function by giving pattern, string line and line number in our text file.
Then dotSearch function generate new pattern without including '.'.
Also get knowledge about where the dot character exists on given pattern. After gathering those data, it calls for KMPSearch_dot function by giving pattern, string line, indexof the place where dot character in given pattern and line number.
KMPSearch_dot function calls for createLpsArray and generate the lps table it needs.
KMPSearch_dot functon is modified to find string patterns on given string according to the explanation of dot regex pattern.

After finding the places of string patterns KMPSearch calls for appendToFile function. Then appendToFile function write that result into a output.txt file.

If there is asterix('*') in given pattern,

It may able to match zero or more occurrences of the preceding character or group.
Main function calls for astrixSearch function by giving pattern, string line and line number in our text file.

For match zero precedence characters:
It generates new pattern without including both '*' and precedence character. Also get knowledge about where the asterix character exist on given pattern.
After gathering those data, asterixSearch calls for KMPSearch function by giving newly generated pattern without both asterix character and precedence character, string line, index of the place where character in given pattern and line number.
KMPSearch function calls for createLpsArray and generate the lps table it needs.

For match one or more precedence characters:
astrixSearch function generate new pattern without including '*' and with including precedence character.
asterixSearch calls for KMPSearch_asterix function by giving newly generated pattern without asterix character and with including precedence character, string line, index of the place where character in given pattern and line number. KMPSearch_astrix function is a modified kmp function to search string patterns with including one or more precedence characters in given pattern.
KMPSearch_astrix function calls for createLpsArray and generate the lps table it needs.

Then with the help of appendToFile function progam write that result into a output.txt file.

If there is plus ('+') in given pattern,

It means our regex program matches one or more occurrences of the preceding character or group.
Now we already have the function, KMPSearch_astrix which can match one or more occurrences of the preceding character or group. Therefor plusSearch function calls for KMPSearch_astrix by giving required arguments. Then KMP_astrix function perfom on selected string line and writes output into output.txt file.

If there is a question ('?') in given pattern,

It means our regex matches zero or one occurrence of the preceding character or group.
Within required aruments our program calls for questionSearch function. questionSearch generates two new patterns.
One is within preceding character.
Other one is within preceding character.
Then questionSearch calls twice KMPSearch with giving two newly generated patterns. At last, it generates the output into output.txt file.

Test Cases to check Regex Patterns

String_Matching_Code includes a text file called testRun.txt for testing regex patterns mentioned above.

- Plain text: Matches any word without special regex characters
    - Test Case 01: bananas

- . (dot): Matches any single character except a newline.
    - Test Case 01: f.x
    - Test Case 02: ar.

- *: Matches zero or more occurrences of the preceding character or group.
    - Test Case 01: cre*k
    - Test Case 01: cre*

- +: Matches one or more occurrences of the preceding character or group.
    - Test Case 01: cre+k
    - Test Case 01: bloo+m

- ?: Matches zero or one occurrence of the preceding character or group.
    - Test Case 01: cre?k
    - Test Case 01: moon?n

W.T.G.Fernando
21000654