Experiment 3.3

Student Name: Pranjal Kumar **UID:** 20BCS3504
Branch: CSE **Section/Group:** 607 /B
Semester: 05 **Date of Performance:** 05/11/2022
Subject Name: Design & Analysis Algorithm **Subject Code:** 20CSP-312

1.AIM

Code and analyze to find all occurrences of a pattern P in a given string S.

2.TASK TO BE DONE

Implementing and analyze to find all occurrences of a pattern P in a given string S.

3.ALGORITHM/FLOWCHART

COMPUTE- PREFIX- FUNCTION (P)

```
1. m ←length [P]           //'p' pattern to be matched
2. Π [1] ← 0
3. k ← 0
4. for q ← 2 to m
5. do while k > 0 and P [k + 1] ≠ P [q]
6. do k ← Π [k]
7. If P [k + 1] = P [q]
8. then k← k + 1
9. Π [q] ← k
10. Return Π
```

KMP-MATCHER (T, P)

```
1. n ← length [T]
2. m ← length [P]
3. Π← COMPUTE-PREFIX-FUNCTION (P)
```

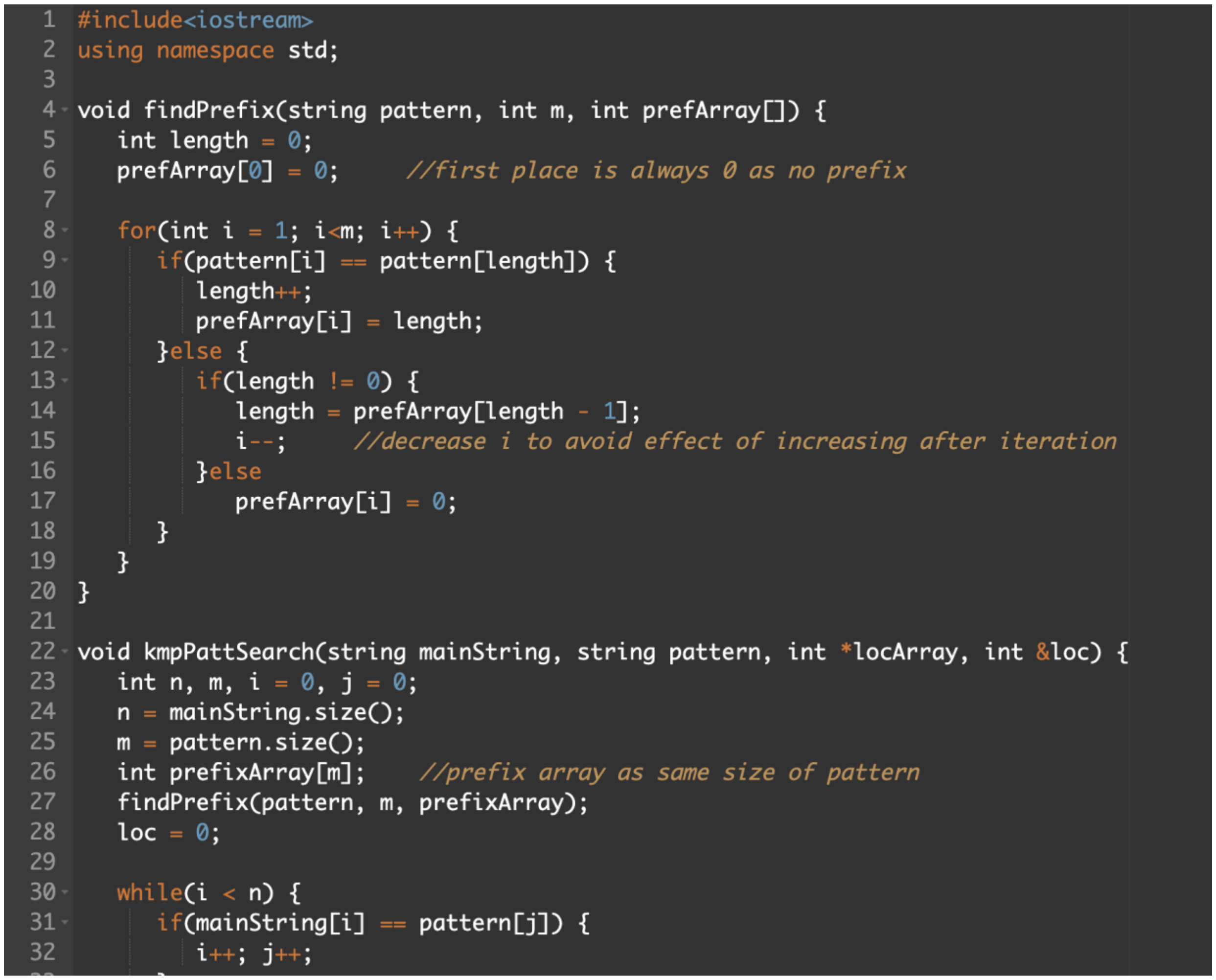
```
4. q ← 0                // numbers of characters matched
5. for i ← 1 to n       // scan S from left to right
6. do while q > 0 and P [q + 1] ≠ T [i]
7. do q ← Π [q]                // next character does not match
8. If P [q + 1] = T [i]
9. then q ← q + 1              // next character matches
10. If q = m                           // is all of p matched?
11. then print "Pattern occurs with shift" i - m
12. q ← Π [q]                                   // look for the next match
```

4.STEPS FOR EXPIREMENT/PRACTICAL/CODE

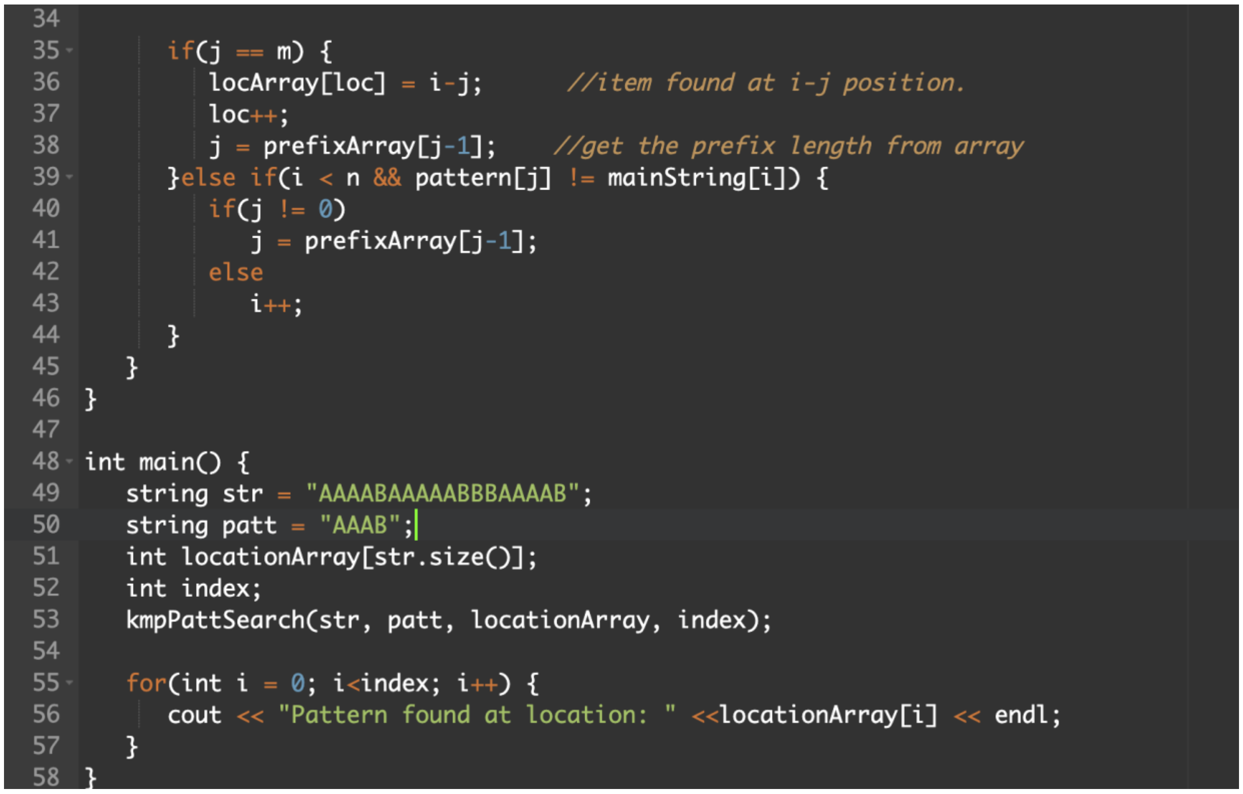
```
#include<iostream>
using namespace std;

void findPrefix(string pattern, int m, int prefArray[]) {
  int length = 0;
  prefArray[0] = 0;     //first place is always 0 as no prefix

  for(int i = 1; i<m; i++) {
    if(pattern[i] == pattern[length]) {
      length++;
      prefArray[i] = length;
    }else {
      if(length != 0) {
        length = prefArray[length - 1];
        i--;     //decrease i to avoid effect of increasing after iteration
      }else
        prefArray[i] = 0;
    }
  }
}
```

```
void kmpPattSearch(string mainString, string pattern, int *locArray, int &loc) {
  int n, m, i = 0, j = 0;
  n = mainString.size();
  m = pattern.size();
  int prefixArray[m];    //prefix array as same size of pattern
  findPrefix(pattern, m, prefixArray);
  loc = 0;

  while(i < n) {
    if(mainString[i] == pattern[j]) {
      i++; j++;
    }

    if(j == m) {
      locArray[loc] = i-j;      //item found at i-j position.
      loc++;
      j = prefixArray[j-1];    //get the prefix length from array
    }else if(i < n && pattern[j] != mainString[i]) {
      if(j != 0)
        j = prefixArray[j-1];
      else
        i++;
    }
  }
}

int main() {
  string str = "AAAABAAAAABBBAAAAB";
  string patt = "AAAB";
  int locationArray[str.size()];
  int index;
```

```
    kmpPattSearch(str, patt, locationArray, index);

    for(int i = 0; i<index; i++) {
      cout << "Pattern found at location: " <<locationArray[i] << endl;
    }
}
```



```
#include<iostream>
using namespace std;

void findPrefix(string pattern, int m, int prefArray[]) {
   int length = 0;
   prefArray[0] = 0;     //first place is always 0 as no prefix

   for(int i = 1; i<m; i++) {
      if(pattern[i] == pattern[length]) {
         length++;
         prefArray[i] = length;
      }else {
         if(length != 0) {
            length = prefArray[length - 1];
            i--;     //decrease i to avoid effect of increasing after iteration
         }else
            prefArray[i] = 0;
      }
   }
}

void kmpPattSearch(string mainString, string pattern, int *locArray, int &loc) {
   int n, m, i = 0, j = 0;
   n = mainString.size();
   m = pattern.size();
   int prefixArray[m];    //prefix array as same size of pattern
   findPrefix(pattern, m, prefixArray);
   loc = 0;

   while(i < n) {
      if(mainString[i] == pattern[j]) {
         i++; j++;
```

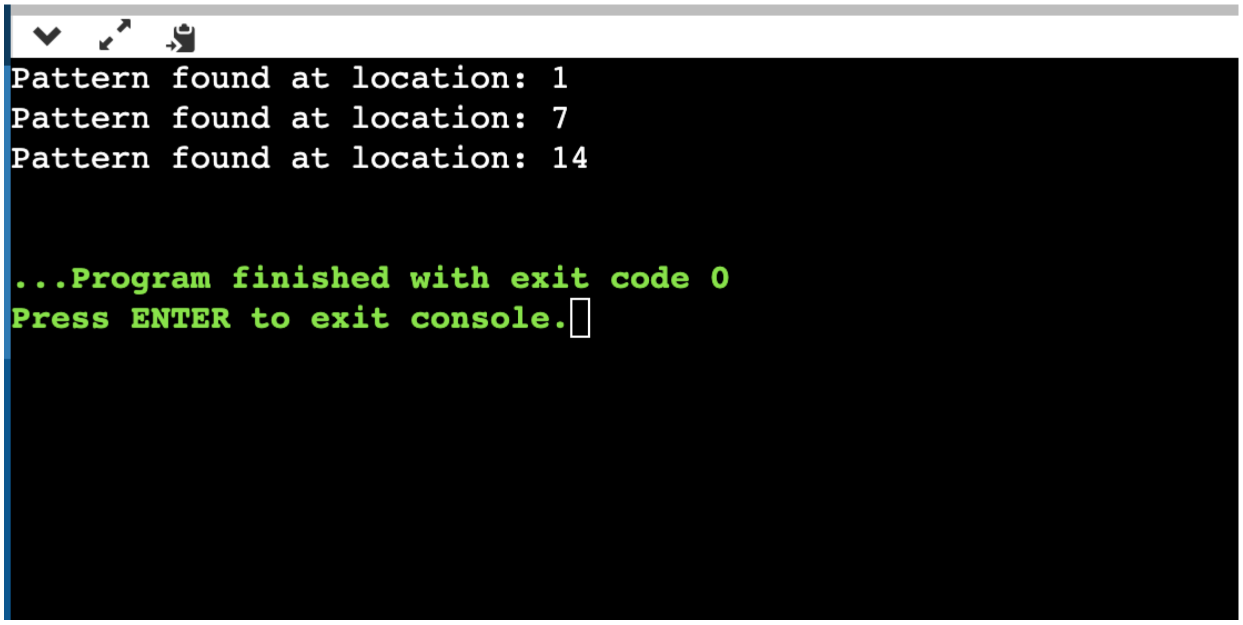


```
        if(j == m) {
            locArray[loc] = i-j;      //item found at i-j position.
            loc++;
            j = prefixArray[j-1];     //get the prefix length from array
        }else if(i < n && pattern[j] != mainString[i]) {
            if(j != 0)
                j = prefixArray[j-1];
            else
                i++;
        }
    }
}

int main() {
    string str = "AAAABAAAAABBBAAAAB";
    string patt = "AAAB";
    int locationArray[str.size()];
    int index;
    kmpPattSearch(str, patt, locationArray, index);

    for(int i = 0; i<index; i++) {
        cout << "Pattern found at location: " <<locationArray[i] << endl;
    }
}
```

4.OBSERVATIONS/DISCUSSIONS/COMPLEXITY ANALYSIS

Time complexity = O(n)

5.OUTPUT/RESULT



```
Pattern found at location: 1
Pattern found at location: 7
Pattern found at location: 14


...Program finished with exit code 0
Press ENTER to exit console.
```

LEARNING OUTCOMES

1.Learnt about algorithm of Knuth Morris Pratt (KMP).

2. complexity of Kmp and Prefix Function.

3. Also learnt about how to analyze the Algorithm.

EVALUATION GRID (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			