Experiment Title- 3.3

Student Name: YANA SRIVASTAVA
UID: 20BCS2279
Branch: CSE
Section/Group: 20BCS-WM-906/B
Semester: 5th
Subject Code: 21CSP-312
Subject Name: DAA LAB

Aim:

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

Algorithm:

COMPUTE-PREFIX-FUNCTION (P)

m = P. length

let π [I .. m] be a new array

π [1] = 0

k = 0

for q = 2 to m

while k > 0 and P [k + l] P [q]

k = π [k]

if P [k + 1] == P [q]

k = k + 1

π [q] = k

return π

KMP-MATCHER(T, P)

n = T.length

m = P. length

π = COMPUTE-PREFIX-FUNCTION (P)

q = 0

for i = 1 to n

while q > 0 and P [q + l] ≠ T [i]

q = π [q]

if P[q + 1] == T [i]

q = q + 1

if q == m

print "Pattern occurs with shift" i - m

q = π [q]

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
			}
			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 = "ANKNANKANNANKAN";
        string patt = "ANKAN";
        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;
        }
}
```

Output:

```
  16:10:08 | user on bridge in ~/Nextcloud/uni/sem5/20CSP-312-daa-lab/experiment-10
→ ./q1
Pattern found at location: 4
Pattern found at location: 10
```

Learning Outcomes:

- Algorithm of Knith Morris Pratt (KMP).
- Complexity of KMP and Prefix function.