CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+ Accredited University

EXPERIMENT – 9

Student Name: Gaganjot Singh **UID: 22BCS14843**
Branch: BE-CSE **Section/Group:** 22BCS_JT_802-B
Semester: 5th **Date of Performance:** 2024
Subject Name: Design & Analysis of Algorithms **Subject Code:** 22CSH-311

1. Aim:
Develop a program and analyze complexity to find the all occurrences of a pattern P in a given string S.

2. Objective:
The objective of this program is to implement a simple pattern matching algorithm to find all occurrences of a given pattern P in a string S. By checking every possible starting position in the string and comparing the subsequent characters with the pattern, the program identifies and outputs the indices where the pattern matches the substring. This basic pattern-matching approach is useful for understanding how substring searching works and serves as a foundation for more advanced algorithms like KMP and Rabin-Karp.

3. Algorithm
- The function findPattern takes two strings, S and P, where S is the main string and P is the pattern.
- It checks for each position in S whether the substring starting at that position matches the pattern P.
- If a match is found, it prints the starting index where the pattern occurs.
- The loop runs from 000 to n−m, where n is the length of S and mmm is the length of P.
- The index of the pattern is printed to terminal.

4. Implementation/Code:

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

void findPattern(string S, string P)
{int n = S.length();
 int m = P.length();

   for (int i = 0; i <= n - m; i++)
   {
      int j;

      for (j = 0; j < m; j++)
      { if (S[i + j] != P[j])
         { break;}}
```

```
        if (j == m)
        {cout << "Pattern found at index " << i << endl;}}}
int main()
{
    string S, P;

    cout << "Enter the main string S: ";
    getline(cin, S);
    cout << "Enter the pattern P: ";
    getline(cin, P);

    findPattern(S, P);
    return 0;}
```

5. Output:

```
PS E:\CU Study\22CSH 311 DAA\codes> cd "e:\CU Study\22CSH 311 DAA\codes\" ; if ($?) { g++ ex9.cpp -o ex9 } ; if ($?) { .
PS E:\CU Study\22CSH 311 DAA\codes> cd "e:\CU Study\22CSH 311 DAA\codes\" ; if ($?) { g++ ex9.cpp -o ex9 } ; if ($?) { .
Enter the main string S: gagan6194daaexperimentgagan682784
Enter the pattern P: gagan6
Pattern found at index 0
Pattern found at index 22
PS E:\CU Study\22CSH 311 DAA\codes>
```

6. Time Complexity :

The time complexity of the above program is O(n×m), where:

- n is the length of the string S (the main text).
- m is the length of the pattern P.

7. Learning Outcomes:

- Learnt how to search for occurrences of a pattern in a given text string using the naive approach.
- Gain experience in iterating through a string and comparing substrings with a given pattern.
- Developed ability to design and implement basic string matching algorithms.
- Understood the time complexity of the naive pattern matching algorithm, O(n×m).
- Learnt how pattern matching can be applied to text processing tasks, such as searching or data analysis.