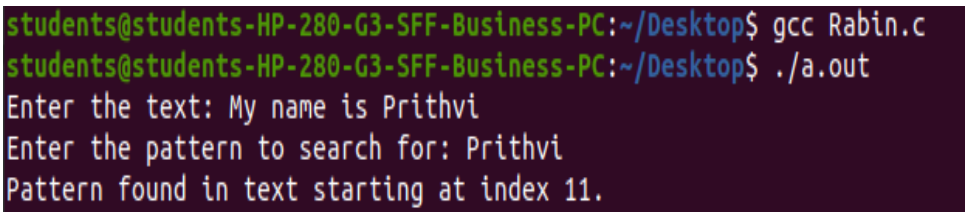
NAME:	Prithvi Singh
UID:	2022301014
SUBJECT	DAA
EXPERIMENT NO:	9
AIM:	To implement Rabin Karp and Naive String Matching Algorithm
Algorithm:	• Rabin Karp Algorithm: 1. n ← length [T] 2. m ← length [P] 3. h ← d^{m-1} mod q 4. p ← 0 5. t_0 ← 0 6. for i ← 1 to m 7. do p ← (dp + P[i]) mod q 8. t_0 ← (dt_0+T [i]) mod q 9. for s ← 0 to n-m 10. do if p = t_s 11. then if P [1.....m] = T [s+1.....s + m] 12. then "Pattern occurs with shift" s 13. If s < n-m 14. t_{s+1} ← (d (t_s-T [s+1]h)+T [s+m+1])mod q

Code:	#include <stdio.h>
	#include <string.h>
	
	#define d 256
	#define q 101
	
	int rabin_karp(char* text, char* pattern) {
	int text_length = strlen(text);
	int pattern_length = strlen(pattern);

```
    int i, j;
    int pattern_hash = 0;
    int text_hash = 0;
    int h = 1;

    for (i = 0; i < pattern_length - 1; i++) {
        h = (h * d) % q;
    }


    for (i = 0; i < pattern_length; i++) {
        pattern_hash = (d * pattern_hash + pattern[i]) % q;
        text_hash = (d * text_hash + text[i]) % q;
    }


    for (i = 0; i <= text_length - pattern_length; i++) {

        if (text_hash == pattern_hash) {

            for (j = 0; j < pattern_length; j++) {
                if (text[i+j] != pattern[j]) {
                    break;
                }
            }


            if (j == pattern_length) {
                return i;
            }
        }
        if (i < text_length - pattern_length) {
            text_hash = (d * (text_hash - text[i] * h) +
text[i+pattern_length]) % q;


            if (text_hash < 0) {
                text_hash += q;
            }
```

```
        }
    }
    return -1;
}
int main() {
    char text[1000], pattern[1000];

    printf("Enter the text: ");
    fgets(text, 1000, stdin);
    printf("Enter the pattern to search for: ");
    fgets(pattern, 1000, stdin);
    text[strcspn(text, "\n")] = 0;
    pattern[strcspn(pattern, "\n")] = 0;
    int result = rabin_karp(text, pattern);
    if (result == -1) {
        printf("Pattern not found in text.\n");
    } else {
        printf("Pattern found in text starting at index %d.\n",
result);
    }
    return 0;
}
```

Output:



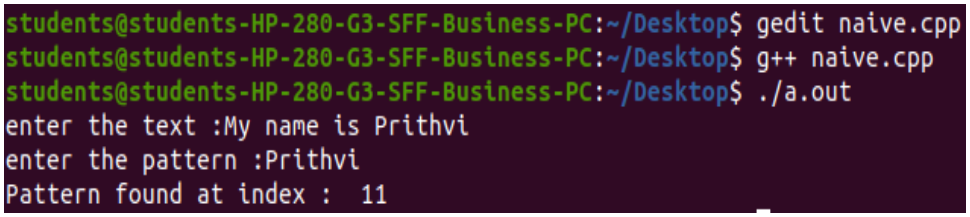
Algorithm:

- **Naive Algorithm:**

1. n ← length [T]
2. m ← length [P]
3. for s ← 0 to n -m
4. do if P [1.....m] = T [s + 1....s + m]

	5. then print "Pattern occurs AT index: "

Code:	#include <iostream>
	#include <string>
	
	using namespace std;
	
	void naiveSearch(string pattern, string text)
	{
	int patternLength = pattern.length();
	int textLength = text.length();
	int i, j;
	
	for (i = 0; i <= textLength - patternLength; i++) {
	for (j = 0; j < patternLength; j++) {
	if (text[i + j] != pattern[j])
	break;
	}
	if (j == patternLength)
	cout << "Pattern found at index : " << i << endl;
	}
	}
	int main()
	{
	string text;
	string pattern;
	
	cout<<"enter the text :";
	getline(cin, text);
	cout<<"enter the pattern :";
	getline(cin, pattern);
	naiveSearch(pattern, text);
	return 0;
	}

Output:	
Conclusion:	Thus we have implemented String matching Algorithm Using Rabin Karp and Naive Algorithm.