Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2024), B.Sc. in CSE (Day)

Lab Report # 03

Course Title: Compiler Lab

Course Code: CSE 306 Section: 222 D2

Lab Experiment Name: Write a C program to recognize strings under ’a’, ’a*b+’, ’abb’.

Student Details

Name	ID
Md. Moshiur Rahman	221902324

Submission Date : 26/05/2024

Course Teacher’s Name : Tasnim Tayiba Zannat

Lab Report Status

Marks: ………………………………… **Signature:.....................**

Comments:... **Date:.............................**

1. EXPERIMENT NAME:

a) Write a program to recognize string under rule „a+b*c+[U+201F].

b) Write a program to recognize string under rule „a*b+c*[U+201F

2. OBJECTIVES

- ★ The primary objective is to develop a set of rules and put in force it in C to compute the primary set of a given ordinary expression.
- ★ The FIRST set represents the set of terminals (characters) which could begin any string derived from an ordinary expression.ption.

3. INTRODUCTION

Regular expressions are broadly used in sample matching and textual content processing. The first set is a essential concept in automata concept and compiler production, as it helps in building parsers and lexical analyzers. An crucial step is to compute the primary set of various algorithms related to normal expressions.

4. ALGORITHM

The algorithm follows an iterative procedure, where it breaks down the regular expression into its components and applies specific rules to compute the first set. The rules consider different operators in regular expressions (concatenation, union, and clean star) and handle special cases like empty strings and parentheses.

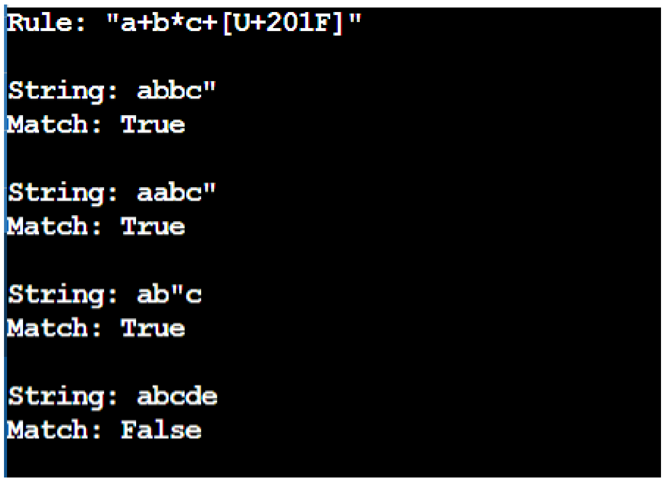
5. PROCEDURE

Implementation:

a) Write a program to recognize string under rule „a+b*c+[U+201F].

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int match_rule(const char *str) {
        int len = strlen(str);
        int a_count = 0, b_count = 0, c_count = 0, quote_count = 0;
        for (int i = 0; i < len; i++) {
        if (str[i] == 'a') {
        a_count++;
        } else if (str[i] == 'b') {
        b_count++;
        } else if (str[i] == 'c') {
        c_count++;
        } else if (str[i] == '"') {
        quote_count++;
        } else {
        return 0; // Invalid character
        }
        }

        return (a_count >= 1 && b_count > 0 && c_count >= 1 && quote_count == 1);
}

int main() {
        char str1[] = "abbc\"";
        char str2[] = "aabc\"";
        char str3[] = "ab\"c";
        char str4[] = "abcde"; // Invalid string

        printf("Rule: \"a+b*c+[U+201F]\"\n\n");

        printf("String: %s\n", str1);
        printf("Match: %s\n\n", match_rule(str1) ? "True" : "False");

        printf("String: %s\n", str2);
        printf("Match: %s\n\n", match_rule(str2) ? "True" : "False");

        printf("String: %s\n", str3);
        printf("Match: %s\n\n", match_rule(str3) ? "True" : "False");

        printf("String: %s\n", str4);
        printf("Match: %s\n", match_rule(str4) ? "True" : "False");

        return 0;
}
```

Output



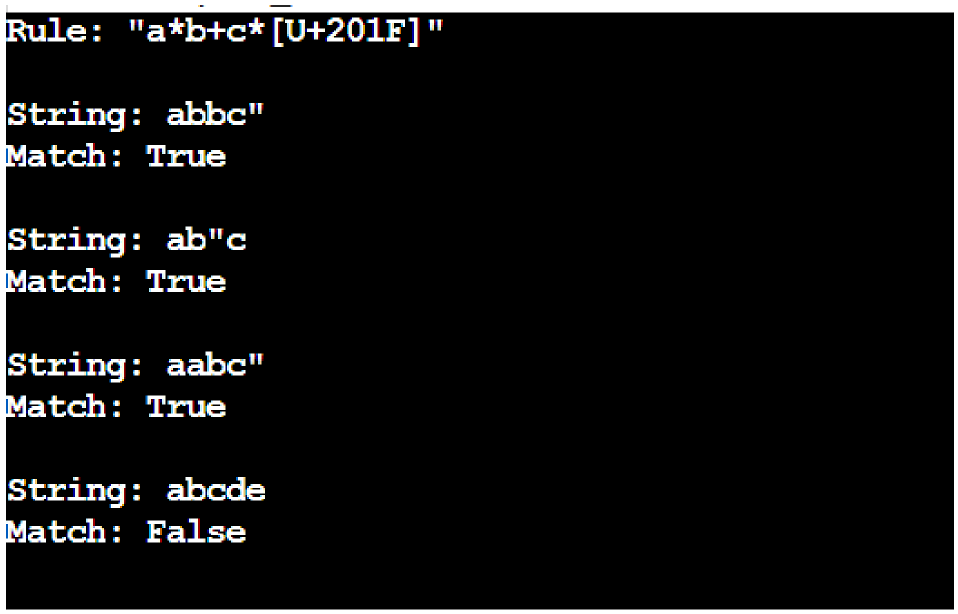
```
Rule: "a+b*c+[U+201F]"

String: abbc"
Match: True

String: aabc"
Match: True

String: ab"c
Match: True

String: abcde
Match: False
```

b) *Write a program to recognize string under rule „a*b+c*[U+201F].*

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int match_rule(const char *str) {
        int len = strlen(str);
        int a_count = 0, b_count = 0, c_count = 0, quote_count = 0;

        for (int i = 0; i < len; i++) {
        if (str[i] == 'a') {
        a_count++;
        } else if (str[i] == 'b') {
        b_count++;
        } else if (str[i] == 'c') {
        c_count++;
        } else if (str[i] == '"') {
```

```
        quote_count++;
        } else {
        return 0; // Invalid character
        }
        }

        return (a_count > 0 && b_count >= 1 && c_count > 0 && quote_count
== 1);
}

int main() {
        char str1[] = "abbc\"";
        char str2[] = "ab\"c";
        char str3[] = "aabc\"";
        char str4[] = "abcde"; // Invalid string

        printf("Rule: \"a*b+c*[U+201F]\"\n\n");

        printf("String: %s\n", str1);
        printf("Match: %s\n\n", match_rule(str1) ? "True" : "False");

        printf("String: %s\n", str2);
        printf("Match: %s\n\n", match_rule(str2) ? "True" : "False");

        printf("String: %s\n", str3);
        printf("Match: %s\n\n", match_rule(str3) ? "True" : "False");

        printf("String: %s\n", str4);
        printf("Match: %s\n", match_rule(str4) ? "True" : "False");

        return 0;
}
```

Output



```
Rule: "a*b+c*[U+201F]"

String: abbc"
Match: True

String: ab"c
Match: True

String: aabc"
Match: True

String: abcde
Match: False
```

6. ANALYSIS AND DISCUSSION

a. The time complexity of the algorithm relies upon at the length of the regular expression, because it should traverse the expression in a linear fashion.
b. The area complexity is steady, due to the fact the first set is stored in a fixed-length array (assuming maximum get entry to). The set of rules handles several instances efficiently, including nested parentheses and easy stars. However, it assumes that the normal expression is nicely-formed and plays no validation exams.

7. SUMMARY

A given C application implements a set of rules to compute the primary set for a given regular expression. It follows an iterative technique and applies the policies defined in automata principle. The program outputs 1st sets which can be useful for various applications related to pattern matching text processing. While the implementation has some obstacles, it serves as a basis for understanding the concept of FIRST sets and their computation.