**Practical 3:**

**Aim: -** Write a program to construct DFA using given regular expression.

**Theory: -**

# **Regular Expression**

* The language accepted by finite automata can be easily described by simple expressions called Regular Expressions. It is the most effective way to represent any language.
* The languages accepted by some regular expression are referred to as Regular languages.
* A regular expression can also be described as a sequence of pattern that defines a string.
* Regular expressions are used to match character combinations in strings. String searching algorithm used this pattern to find the operations on a string.

**For instance:**

In a regular expression, x\* means zero or more occurrence of x.

It can generate {e, x, xx, xxx, xxxx, .....}

In a regular expression, x+ means one or more occurrence of x.

It can generate {x, xx, xxx, xxxx, .....}

## **Operations on Regular Language**

The various operations on regular language are:

* **Union:** If L and M are two regular languages then their union L U M is also a union.
  + - * L U M = {s | s is in L or s is in M}
* **Intersection:** If L and M are two regular languages then their intersection is also an intersection.
  + - * L ⋂ M = {st | s is in L and t is in M}
* **Kleen closure:** If L is a regular language then its Kleen closure L1\* will also be a regular language.
  + - * L\* = Zero or more occurrence of language L.

### Example 1:

Write the regular expression for the language accepting all combinations of a's, over the set ∑ = {a}

**Solution:**

All combinations of a's means a may be zero, single, double and so on. If a is appearing zero times, that means a null string. That is we expect the set of {ε, a, aa, aaa, ....}. So we give a regular expression for this as:

1. R = a\*

That is Kleen closure of a.

### Example 2:

Write the regular expression for the language accepting all combinations of a's except the null string, over the set ∑ = {a}

**Solution:**

The regular expression has to be built for the language

1. L = {a, aa, aaa, ....}

This set indicates that there is no null string. So we can denote regular expression as:

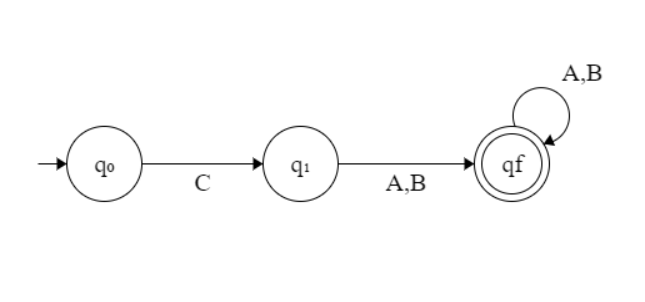
R = a+

**Output form**

Given a string S, the task is to design a Deterministic Finite Automata (DFA) for accepting the language L = C (A + B)+. If the given string is accepted by DFA, then print “Yes”. Otherwise, print “No”.

Input: S = “CABABABAB”  
Output: Yes  
Explanation: The given string is of the form C(A + B)+ as the first character is C and it is followed by A or B.

Input: S = “ACCBBCCA”  
Output: No



* If the given string is of length less than equal to 1, then print “No”.
* If the first character is always C, then traverse the remaining string and check if any of the characters is A or B.
* If there exists any character other than A or B while traversing in the above step, then print “No”.
* Otherwise, print “Yes”.
* Below is the implementation of the above approach:

**Program:-**

# Function to find whether the given

# is Accepted by the DFA

def DFA(str, N):

# If n <= 1, then prNo

if (N <= 1):

print("No")

return

# To count the matched characters

count = 0

# Check if the first character is C

if (str[0] == 'C'):

count += 1

# Traverse the rest of string

for i in range(1, N):

# If character is A or B,

# increment count by 1

if (str[i] == 'A' or str[i] == 'B'):

count += 1

else:

break

else:

# If the first character

# is not C, pr-1

print("No")

return

# If all characters matches

if (count == N):

print("Yes")

else:

print("No")

# Driver Code

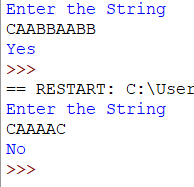
if \_\_name\_\_ == '\_\_main\_\_':

str = "CAABBAAB"

N = len(str)

DFA(str, N)

**Output:-**



**Conclusion**:-

The given program Successfully checks whether the given string is DFA or not.

**References:-**

<https://www.javatpoint.com/automata-regular-expression>

https://www.geeksforgeeks.org/program-to-construct-dfa-for-regular-expression-c-a-bQ