**NAME- DEVANGI PURKAYASTHA**

**REG NO- RA1811003010689**

**EXP -3 : CONVERSION OF REGULAR EXPRESSION TO NFA**

**DATE: 12 Feb, 2021**

**CODE:**

if \_\_name\_\_ == '\_\_main\_\_':

    states = {

    "1": "0 a 1",

    "2": "0 e 1 a 2 e 5 0 e 3 b 4 e 5",

    "3": "0 a 1 b",

    "4": "0 e 1 a 2 e 3 0 e 3 2 e 1"

    }

    choices = ["1. a", "2. a/b", "3. ab", "4. a\*"]

    while(True):

        print("enter your choice")

        for i in choices:

            print(i)

        choice = input()

        if(choice == "1"):

            print(states["1"])

        elif(choice == "2"):

            print(states["2"])

        elif(choice == "3"):

            print(states["3"])

        elif(choice == "4"):

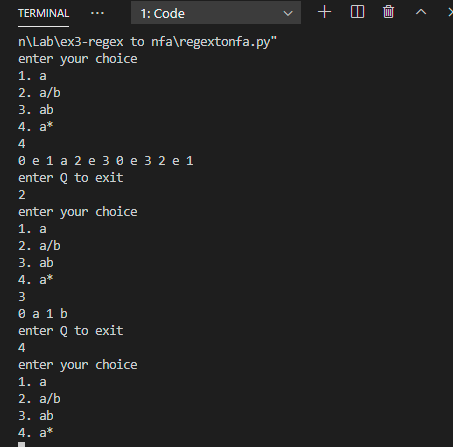
            print(states["4"])

        print("enter Q to exit")

        if input() == "Q":

            break

**OUTPUT:**



**ALGORITHM:**

The algorithm works [recursively](https://en.wikipedia.org/wiki/Recursively) by splitting an expression into its constituent subexpressions, from which the NFA will be constructed using a set of rules.[[3]](https://en.wikipedia.org/wiki/Thompson%27s_construction#cite_note-3) More precisely, from a regular expression *E*, the obtained automaton *A* with the transition function which respects the following properties:

* *A* has exactly one initial state *q*0, which is not accessible from any other state. That is, for any state *q* and any letter *a*, {\displaystyle \Delta (q,a)} does not contain *q*0.
* *A* has exactly one final state *qf*, which is not co-accessible from any other state. That is, for any letter *a*, {\displaystyle \Delta (q\_{f},a)=\emptyset }.
* Let *c* be the number of concatenation of the regular expression *E* and let *s* be the number of symbols apart from parentheses — that is, |, \*, *a* and *ε*. Then, the number of states of *A* is 2*s* − *c* (linear in the size of *E*).
* The number of transitions leaving any state is at most two.
* Since an NFA of *m* states and at most *e* transitions from each state can match a string of length *n* in time *O*(*emn*), a Thompson NFA can do pattern matching in linear time, assuming a fixed-size alphabet.

**RESULT**: The given program for conversion of a regular expression to NFA has been successfully executed