18CSC207J-Advance Programming Practice - Structured Programming – Lab Programs

Symbolic ,Logical, functional and automata Programming Paradigm

Name :- Puneet Sharma
Reg. No. :- RA1911003010331
Class :-CSE F1

SYMBOLIC PROGRAMMING

1. Solve the following using symbolic paradigm:

i. Calculate sqrt (2) with 100 decimals.

ii. Calculate (1/2+1/3) in rational arithmetic.

iii. Calculate the expanded form of (x+y) ^ 6.

iv. Simplify the trigonometric expression sin (x) / cos (x)

v. Calculate sin x -xx^3n

In [11]:
```
from sympy import *
#i) Calculate sqrt (2) with 100 decimals
print(N(sqrt(2)*1,100))
```

```
1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753432764157 3
```

In [5]:
```
#ii) Calculate (1/2+1/3) in rational arithmetic
a=Rational(1,2)
b=Rational(1,3)
print(a+b)
```

```
5/6
```

In [9]:
```
#iii) Calculate the expanded form of (x+y) ^ 6.
import sympy as sym
x = sym.Symbol('x')
y = sym.Symbol('y')
sym.expand((x+y)**6)
```

Out[9]: $x^6 + 6x^5y + 15x^4y^2 + 20x^3y^3 + 15x^2y^4 + 6xy^5 + y^6$

In [12]:
```
#iv) Simplify the trigonometric expression sin (x) / cos (x)
print(simplify(sin(x)/cos(x)))
```

```
tan(x)
```

In [8]:
```
#v) Calculate sinx-xx^3n
print(simplify(sin(x)-x**3*n))
```

```
-n*x**3 + sin(x)
```

2. Develop a python code for to carryout the operations on the given algebraic manipulation for the given expression a2−ab+ab−b2=a2−b2 by using the symbolic programming paradigms principles.

In [5]:
```
#Q2
import sympy as sym
a = sym.Symbol('a')
b=sym.Symbol('b')
sym.simplify(a**2 -a*b +a*b -b**2)
```

Out[5]: $a^2 - b^2$

3. Give the Symbolic program for the expression given below:

a. a2 da

b. 2x+y2 c. 1/10 + 1/5

d. d/dx(sin(x))

In [4]:
```
#i) integrate a^2
import sympy as sym
a = sym.Symbol('a')
b=sym.Symbol('b')
sym.integrate(sym.integrate(a**2,a))
```

Out[4]: $\frac{a^4}{12}$

In [2]:
```
#ii) 2x+y^2
import sympy as sym
x = sym.Symbol('x')
y = sym.Symbol('y')
sym.simplify(2*x + y**2)
```

Out[2]: $2x + y^2$

In [3]:
```
#iii) 1/10 + 1/5
sym.Rational(1,10)+sym.Rational(1,5)
```

Out[3]: $\frac{3}{10}$

In [7]:
```
#iv) d/dx(sin(x))
x = symbols('x')
print(diff(sin(x)))
```

```
cos(x)
```

Logic Programming

1. Implement using pyDatalog:

Assume given a set of facts of the form father(name1,name2) (name1 is the father of name2).

a. Define a predicate brother(X,Y) which holds iff X and Y are brothers.

b. Define a predicate cousin(X,Y) which holds iff X and Y are cousins.

c. Define a predicate grandson(X,Y) which holds iff X is a grandson of Y. d. Define a predicate descendent(X,Y) which holds iff X is a descendent of Y. e. Consider the following genealogical tree:

a

/ \

b c

/ \ |

d e f

What are the answers generated by your definitions for the queries:

brother(X,Y)

cousin(X,Y)

grandson(X,Y)

descendent(X,Y)

In [1]:
```
#Q1
from pyDatalog import pyDatalog
pyDatalog.create_terms('a,b,c,d,e,f,brother,cousin,grandson,descendent,X,Y')
+brother('b','c')
+brother('d','e')
+cousin('d','f')
+cousin('e','f')
+grandson('d','a')
+grandson('e','a')
+grandson('f','a')
+descendent('b','a')
+descendent('c','a')
+descendent('d','b')
+descendent('f','c')
print(pyDatalog.ask('brother(X,Y)'))
print(pyDatalog.ask('cousin(X,Y)'))
print(pyDatalog.ask('grandson(X,Y)'))
print(pyDatalog.ask('descendent(X,Y)'))
```

```
{('b', 'c'), ('d', 'e')}
{('e', 'f'), ('d', 'f')}
{('f', 'a'), ('e', 'a'), ('d', 'a')}
{('d', 'b'), ('c', 'a'), ('b', 'a'), ('f', 'c')}
```

2. Encode the following facts and rules in pyDatalog: Bear is big

- Elephant is big
- Cat is small Bear is brown
- Cat is black
- Elephant is gray
- An animal is dark if it is black
- An animal is dark if it is brown

Write a query to find which animal is dark and big.

```
In [1]: #q2
from pyDatalog import pyDatalog
pyDatalog.create_terms('X,Y,Z,bear,elephant,cat,small,big,brown,black,gray,dark')
+big('elephant')
+big('bear')
+small('cat')
+black('cat')
+brown('bear')
+gray('elephant')
dark(X)<=black(X) or brown(X)
print(big(X),dark(X))
```

```
X
--------
bear
elephant X
---
cat
```

3. The following are the marks scored by 5 students.

Student Name Mark

Ram 90

Raju 45

Priya 85

Carol 70

Shyam 80

Enter the above data using pyDatalog. Write queries for the following:

a. Print Student name and mark of all students.

b. Who has scored 80 marks?

c. What mark has been scored by Priya?

d. Write a rule ‘passm’ denoting that pass mark is greater than 50. Use the rule to print all students who failed.

e. Write rules for finding grade letters for a marks and use the rule to find the grade letter of a given mark.

In [4]:

```
#Q3
from pyDatalog import pyDatalog
pyDatalog.create_terms('X,Y,Z,student,marks,passm,grades')
+student('Ram')
+student('Raju')
+student('Priya')
+student('Carol')
+student('Shyam')
+marks('90','Ram')
+marks('45','Raju')
+marks('85','Priya')
+marks('70','Carol')
+marks('80','Shyam')
+grades('Ram','O')
+grades('Priya','A')
+grades('Shyam','A')
+grades('Carol','B')
+grades('Raju','E')
print(marks(X,Y))
print(marks('80',X))
print(marks(X,'Priya'))
passm(X)<=grades(X,'E')
print(passm(X))
```

```
X  | Y
---|------
80 | Shyam
70 | Carol
85 | Priya
45 | Raju
90 | Ram
80 | shyam
70 | carol
85 | priya
45 | raju
90 | ram
X
-----
Shyam
shyam
X
--
85
X
----
raju
Raju
```

4. Solve the set of queries in the previous question using imperative programming paradigm in Python. Store the data in a dictionary.

In [6]:
```
#Q4
marks={90:'Ram',85:'Priya',80:'Shyam',70:'Carol',45:'Raju'}
for i in marks:
    print(i,marks[i])
print(marks[80])
for i in marks:
    if marks[i]=='priya':
        print(i)
for i in marks:
    if i<50:
        print(marks[i])
for i in marks:
    if i>=90:
        print(marks[i],'O')
    elif i<90 and i>=80:
        print(marks[i],'A')
    elif i<80 and i>=70:
        print(marks[i],'B')
    elif i<70 and i>=60:
        print(marks[i],'C')
    elif i<60 and i>=50:
        print(marks[i],'D')
    else:
        print(marks[i],'E')
```

```
90 Ram
85 Priya
80 Shyam
70 Carol
45 Raju
Shyam
Raju
Ram O
Priya A
Shyam A
Carol B
Raju E
```

5. Write a recursive program to find factorial of a number using pyDatalog.

In [2]:
```
#Q5
from pyDatalog import pyDatalog
pyDatalog.create_terms('factorial, N')
num=int(input('Enter any number:'))
factorial[N] = N*factorial[N-1]
factorial[1] = 1
print(factorial[num]==N)
```

```
Enter any number:8
N
-----
40320
```

Functional Programming

1. Calculate the following using Lambda calculus:

   a. T AND F
   b. 3 * 4

```
1. Calculate the following using Lambda calculus:
a. T AND F
b. 3 * 4
```

In [1]:
```
y=lambda s: s and False
print("T AND F = ",y(True))
x = lambda a : a*4
print("3*4= ",x(3))
print("\n")
```

```
T AND F =  False
3*4=  12
```

2. Lambda functions

a. Write a lambda function to convert measurements from meters to feet.

b. Write a lambda function in Python to implement the following lambda expression:

$$(\lambda f.\ \lambda m.\ (f + m)a)(\lambda x.\ x2)(b)$$

Note: You need to write a nested lambda function for implementing f+m where f takes the square function (which takes argument x) passed as a parameter. The above expression calculates a2+b.

In [ ]:
```
#Q2 (A)
feet = lambda m: m*(3.281)
meters = int(input("Enter the number of meters to be converted: "))
print("{:0.2f} feet".format(feet(meters)))
```

In [48]:
```
#Q2 (B)
square = lambda x: x**2
total = lambda f, b: lambda a: f(a)+b
a = int(input("Enter value for a: "))
b = int(input("Enter value for b: "))
print(total(square, b)(a))
```

```
Enter value for a: 5
Enter value for b: 10
35
```

3. Passing and returning a function as an argument

Define a function ‘square’ for squaring a number. Define a function named ‘twice’ that takes a function f as an argument and returns f(f(x)). Using ‘twice’ and ‘square’ create a function ‘quad’ that takes n as an argument and returns n4 ‘quad’ should not be defined explicitly. It should only be created as a variable which is then assigned a function.

In [20]:
```
#Q3
def square(number):
    value = number * number
    return value
def twice(func,num):
    return func(func(num))
n=int(input('Enter a number:'))
k=twice(square,n)
print(k)
```

```
Enter a number:5
625
```

4. Closure

A Closure is a function object that remembers values in enclosing scopes even if they are not present in memory. We have a closure in Python when a nested function references a value in its enclosing scope.

a. Study the following program by executing it:

```
def multiplier_of(n):

def multiplier(number):

return number*n

return multiplier

multiplywith5 = multiplier_of(5)

print(multiplywith5(9))
```

b. In a lottery system, random number is chosen by retrieving the number from a random index from a list of random numbers. Write a program to choose a random number in this way. You must use nested functions – the inner function chooses a number from a random index and the outer function generates a random list of numbers. The outer function takes n as a parameter where is the maximum number that can be put in the random list.

In [54]:
```
#Q4 (A)
def multiplier_of(n):
    def multiplier(number):
            return number*n
    return multiplier
multiplywith5 = multiplier_of(5)
print(multiplywith5(9))
```

```
45
```

In [2]:
```
#Q4 (B)
import random
m=10
print("Let the maximum limit for generating random numbers be "+str(m))
lst=[]
for i in range(m):
    lst.append(random.randint(1,m))
for j in range(1):
    d1=random.randrange(0,m,1)
    print("Index of the random number is ",lst[d1])
print("\n")
```

```
Let the maximum limit for generating random numbers be 10
Index of the random number is  10
```

6. Map

A secret message needs to be sent. Use the map function to encrypt the message using Caesar cipher.

In [3]:
```
#Q6
def encrypt(letter, s):
    if (letter.isupper()):
        return chr((ord(letter) + s - 65) % 26 + 65)
    elif (letter.islower()):
        return chr((ord(letter) + s - 97) % 26 + 97)
    else:
        return letter
text = input("Enter the word: ")
s = int(input("Enter the shift: "))
shift = [s] * len(text)
result = list(map(encrypt, text, shift))
print("Encrypted text: ", end="")
for i in result:
 print(i, end="")
```

```
Enter the word: my name is puneet sharma
Enter the shift: 4
Encrypted text: qc reqi mw tyriix wlevqe
```

7. Reduce

Given runs scored by 2 players in a series of matches, write a Python program using reduce function to find who is the better player of the two in terms of maintaining consistency. (You need to find SD).

In [37]:
```
#7
import functools
import operator
player1 = [100, 20, 50, 66, 72, 32]
player2 = [56, 65, 78, 45, 33, 69]
mean1 = (functools.reduce(operator.add, player1)) / len(player1)
mean2 = (functools.reduce(operator.add, player2)) / len(player2)
m1 = [mean1] * len(player1)
m2 = [mean2] * len(player2)
def variance(p, m):
    var = p - m
    return var**2

sqvar1 = list(map(variance, player1, m1))
sqvar2 = list(map(variance, player2, m2))
var1 = (functools.reduce(operator.add, sqvar1)) / len(sqvar1)
var2 = (functools.reduce(operator.add, sqvar2)) / len(sqvar2)
sd1 = var1**0.5
sd2 = var2**0.5
if sd1 == sd2:
    print("Both players are consistent.")
elif sd1 > sd2:
    print("Player 2 is more consistent.")
else:
    print("Player 1 is more consistent.")
```

```
Player 2 is more consistent.
```

8. Filter

The marks scored by a class of students in 5 different subjects are stored in a list of lists. Using the filter function, write a program to find the students who failed in one or more subjects.

In [38]:
```
#Q8
marks = [[88, 77, 66, 77, 99],[56, 78, 31, 35, 54],[53, 71, 64, 56, 14]]
def fail(marks):
    for i in marks:
        if i < 40:
            return True
result = filter(fail, marks)
res = list(result)
for i in range(0, len(marks)):
    if marks[i] in res:
        print("Student #{} failed in at least one subject.".format(i+1))
```

```
Student #2 failed in at least one subject.
Student #3 failed in at least one subject.
```

9. Map + reduce + filter

Given two trending topics and a bunch of tweets, write a Python program to count the number of tweets that contain each topic. You need to do this by putting together map(), reduce() and filter() functions.

In [4]:
```
#Q9
import functools
import operator
tweets = ["First ever player to score 6000 runs in #IPL - Just another day at the ‘office’ for #ViratKohli!",
 "My @rajasthanroyals outfit has arrived! Now to be well enough to wear it....Thanks @IamSanjuSamson & the RR management! All the
 "How amazing that both the centuries in this year’s #IPL have been scored by Malayalis, when Kerala has so long been regarded as
 "Most 10-wicket wins in IPL: 4 - RCB, 2 - MI, 2 - SRH, 2 - CSK #RCBvsRR #RCB #RR #IPL #IPL2021 #CricTracker",
 "India would have never eliminated polio, if people had to BUY polio vaccines. Same logic applies for #COVID19",
 "The Ontario Science Table @COVIDSciOntario has recommended that 50% of all available vaccines go to hotspot neighbourhoods with
 "UPA home minister resigned after terror attack on Mumbai. Why doesn’t NDA health minister resign for this #COVID19 mess? I woul
 "Khalsa Aid India arranges Oxygen concentrators for COVID-19 patients, will deliver them in Delhi. #COVID19"]
ipl = ["#ipl"] * len(tweets)
covid = ["#covid19"] * len(tweets)
def segregate_tweets(lst, topic):
    if topic in lst.lower():
        return lst
resultipl = list(map(segregate_tweets, tweets, ipl))
resultcovid = list(map(segregate_tweets, tweets, covid))
def filter_list(lst):
    if lst != "":
        return lst
result_ipl = list(filter(filter_list, resultipl))
result_covid = list(filter(filter_list, resultcovid))
print("IPL tweets: {}".format(len(result_ipl)))
print("COVID19 tweets: {}".format(len(result_covid)))
```

```
IPL tweets: 4
COVID19 tweets: 4
```

AUTOMATA PROGRAMMING:

DFA:

1. Write a automata code for the Language that accepts all and only those strings that contain 001

2. Write a automata code for L(M) ={ w | w has an even number of 1s}

3. Write a automata code for L(M) ={0,1}*

4. Write a automata code for L(M)=a + aa*b. 5. Write a automata code for L(M)={(ab)n| n □ N}

6. Write a automata code for Let Σ = {0, 1}.Given DFAs for {}, {ε}, Σ*, and Σ+.

In [1]:
```
#1st
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q0'},
        'q1': {'0': 'q2', '1': 'q0'},
        'q2': {'0': 'q2', '1': 'q3'},
        'q3': {'0': 'q3', '1': 'q3'}
    },
    initial_state='q0',
    final_states={'q3'}
)
for i in range(1,4):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :0001
Accepted
Enter the string :1001
Accepted
Enter the string :1011
Rejected
```

In [2]:
```
#2nd
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'0': 'q1', '1': 'q2'},
        'q2': {'0': 'q2', '1': 'q1'}
    },
    initial_state='q0',
    final_states={'q2'}
)
for i in range(1,4):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :1111
Accepted
Enter the string :0101
Accepted
Enter the string :1110
Rejected
```

In [3]:
```
#3rd
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q0'}
    },
    initial_state='q0',
    final_states={'q0'}
)
for i in range(1,8):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :01
Accepted
Enter the string :001
Accepted
Enter the string :0
Accepted
Enter the string :0011
Accepted
Enter the string :1001
Accepted
Enter the string :011
Accepted
Enter the string :10011
Accepted
```

In [4]:
```
#4th
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': 'q1', 'b': 'q5'},
        'q1': {'a': 'q2', 'b': 'q5'},
        'q2': {'a': 'q3', 'b': 'q4'},
        'q3': {'a': 'q2', 'b': 'q5'},
        'q4': {'a': 'q5', 'b': 'q5'},
        'q5': {'a': 'q5', 'b': 'q5'}
    },
    initial_state='q0',
    final_states={'q1', 'q4'}
)
for i in range(1,6):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :a
Accepted
Enter the string :ab
Rejected
Enter the string :aab
Accepted
Enter the string :aaaab
Accepted
Enter the string :baaaab
Rejected
```

In [5]:
```
#5th
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': 'q1', 'b': 'q3'},
        'q1': {'a': 'q3', 'b': 'q2'},
        'q2': {'a': 'q1', 'b': 'q3'},
        'q3': {'a': 'q3', 'b': 'q3'}
    },
    initial_state='q0',
    final_states={'q2'}
)
for i in range(1,6):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :ab
Accepted
Enter the string :abab
Accepted
Enter the string :baba
Rejected
Enter the string :aaabbb
Rejected
Enter the string :ababab
Accepted
```

In [6]:
```python
#6.1
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q0'},
        'q1': {'0': 'q1', '1': 'q1'}
    },
    initial_state='q0',
    final_states={'q1'}
)
for i in range(1,8):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :1
Rejected
Enter the string :01
Rejected
Enter the string :
Rejected
Enter the string :001
Rejected
Enter the string :00011
Rejected
Enter the string :00011
Rejected
Enter the string :0011
Rejected
```

In [7]:
```python
#6.2
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q1'},
        'q1': {'0': 'q1', '1': 'q1'}
    },
    initial_state='q0',
    final_states={'q0'}
)
for i in range(1,6):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :00
Rejected
Enter the string :101
Rejected
Enter the string :1011
Rejected
Enter the string :
Accepted
Enter the string :100101
Rejected
```

In [8]:
```
#6.3
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q0'}
    },
    initial_state='q0',
    final_states={'q0'}
)
for i in range(1,8):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :00
Accepted
Enter the string :011
Accepted
Enter the string :101
Accepted
Enter the string :
Accepted
Enter the string :11010
Accepted
Enter the string :1100
Accepted
Enter the string :001
Accepted
```

In [9]:
```
#6.4
from automata.fa.dfa import DFA
dfa = DFA(
    states={'q0', 'q1'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q1'},
        'q1': {'0': 'q1', '1': 'q1'}
    },
    initial_state='q0',
    final_states={'q1'}
)
for i in range(1,8):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :
Rejected
Enter the string :01
Accepted
Enter the string :1001
Accepted
Enter the string :1101
Accepted
Enter the string :1111
Accepted
Enter the string :0
Accepted
Enter the string :01
Accepted
```

NFA:

1. Write a automata code for the Language that accepts all end with 01

2. Write a automata code for L(M)= a + aa*b + a*b. 3. Write a automata code for Let Σ = {0,1}.

Given NFAs for {}, {ε}, {(ab)n | n □ N}, which has regular expression (ab)*.

In [4]:
```
from automata.fa.nfa import NFA
nfa = NFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': {'q1','q0'}, '1': {'q0'}},
        'q1': {'1': {'q2'}},
        'q2': {}
    },
    initial_state='q0',
    final_states={'q2'}
)
for i in range(1,4):
    num = input("Enter the string :")
    if(nfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :0101
Accepted
Enter the string :0001
Accepted
Enter the string :0111
Rejected
```

In [2]:
```
from automata.fa.nfa import NFA
nfa = NFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': {'q1','q2'}},
        'q1': {'a': {'q2','q4'}, 'b': {'q4'}},
        'q2': {'a': {'q2'}, 'b': {'q3'}},
        'q3': {},
        'q4': {}
    },
    initial_state='q0',
    final_states={'q1','q3'}
)
for i in range(1,6):
    num = input("Enter the string :")
    if(nfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :a
Accepted
Enter the string :ab
Accepted
Enter the string :aab
Accepted
Enter the string :abab
Rejected
Enter the string :aaa
Rejected
```

In [3]:
```
from automata.fa.nfa import NFA
nfa = NFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': {'q1'}},
        'q1': {'b': {'q0', 'q2'}},
        'q2': {}
    },
    initial_state='q0',
    final_states={'q2'}
)
for i in range(1,8):
    num = input("Enter the string :")
    if(nfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

```
Enter the string :ab
Accepted
Enter the string :abab
Accepted
Enter the string :abababab
Accepted
Enter the string :aabb
Rejected
Enter the string :aaa
Rejected
Enter the string :baba
Rejected
Enter the string :aba
Rejected
```