NAME: SHASHANK TRIPATHI RA2011031010096 CSE-IT (O2)

WEEK 7: SYMBOLIC, AUTOMATA, LOGICAL PROGRAMMING PARADIGM

AIM: To implement various aspects of symbolic, automata, logical programming paradigm.

Symbolic programming paradigm

```
#Code Q1
import sympy as sym
print((sym.sqrt(2)).evalf(100))
```

```
1.414213562373095048801688724209698078569671875376948073176679737990732478462107038850387534327641573
```

```
[2] #Code Q2
import sympy as sym
print(sym.Rational(1,2)+sym.Rational(1,3))
```

```
5/6
```

```
[4] #Code Q3
import sympy as sym
x=sym.Symbol('x')
y=sym.Symbol('y')
print(sym.expand((x+y)**6))
```

```
x**6 + 6*x**5*y + 15*x**4*y**2 + 20*x**3*y**3 + 15*x**2*y**4 + 6*x*y**5 + y**6
```

```
[5] #Code Q4
from sympy import *
x = symbols('x')
print(trigsimp(sin(x) / cos(x)))
```

```
tan(x)
```

```
[6] #Code Q5
from sympy import *
x = symbols('x')
expr = (sin(x)-x)/x**3;
limit_expr = limit(expr, x, 0)
print(limit_expr)
```

```
-1/6
```

```
[7] #Code Q6
from sympy import *

x = symbols('x')
expr = log(x)
print("Expression : {} ".format(expr))
expr_diff = Derivative(expr, x)
print("Derivative of expression with respect to x : {}".format(expr_diff))
print("Value of the derivative : {} ".format(expr_diff.doit()))
```

```
Expression : log(x)
Derivative of expression with respect to x : Derivative(log(x), x)
Value of the derivative : 1/x
```

```
[9]  #Code Q7
     import sympy as sp
     x, y = sp.symbols('x, y')
     eq1 = sp.Eq(x + y , 2)
     eq2 = sp.Eq(2*x + y , 0)
     ans = sp.solve((eq1, eq2), (x, y))
     print(ans)
```

```
{x: -2, y: 4}
```

```
[10] #Code Q8
     from sympy import *
     x = symbols('x')
     exp = sin(x)

     print("Before Integration : {}".format(exp))

     # Use sympy.integrate() method
     intr = integrate(exp, x)

     print("After Integration : {}".format(intr))
```

```
Before Integration : sin(x)
After Integration : -cos(x)
```

```
[13] #Code Q9
     from sympy import *
     import sympy as sym
     x = symbols('x')
     f=sym.Function('f')(x)
     ans=sym.dsolve(f.diff(x,x)+9*f,0)
     print(ans)
```

```
Eq(f(x), C1*sin(3*x) + C2*cos(3*x))
```

```
#Code Q10
import numpy as np
A=np.array([[3,7],[4,-2]])
B=np.array([12,5])
print(np.linalg.solve(A,B))
```

```
[1.73529412 0.97058824]
```

LOGICAL PROGRAMMING PARADIGM

1

```
from pyDatalog import pyDatalog as py
py.create_terms("brother,father,cousin,grandson,descendent,X,Y,Z,W,a,b,c,d,e,f")
+father('a','b')
+father('a','c')
+father('b','d')
+father('b','e')
+father('c','f')
brother(X,Y) <= (father(Z,X)) & (father(Z,Y)) & ~(X==Y)
cousin(X,Y) <= (father(Z,X)) & (father(W,Y)) & (brother(Z,W))
grandson(X,Y)<= (father(Y,Z)) & (father(Z,X))
descendent(X,Y) <= (father(Y,X))
descendent(X,Y) <= (father(Z,X)) & (descendent(Z,X))
print(brother(X,Y))
print(cousin(X,Y))
print(grandson(X,Y))
print(descendent(X,Y))
```

```
X | Y
--|--
e | d
d | e
c | b
b | c
X | Y
--|--
f | e
f | d
d | f
e | f
X | Y
--|--
f | a
e | a
d | a
X | Y
--|--
b | a
c | a
d | b
e | b
f | c
```

2

```
from pyDatalog import pyDatalog
pyDatalog.create_terms('Bear,Elephant,Cat,Black,Gray,Brown,Big,small,size,color,X
+size("Bear","Big")
+size("Elephant","Big")
+size("Cat","small")
+color("Bear","Brown")
+color("Cat","Black")
+color("Elephant","Gray")
dark(X) <= (color(X,"Black"))
dark(X) <= (color(X,"Brown"))
print(dark(X) &(size(X,'Big')))
```

```
X
----
Bear
```

3

```
from pyDatalog import pyDatalog
pyDatalog.create_terms("N,Factorial")
Factorial[N]=N*Factorial[N-1]
Factorial[1]=1
print(Factorial[4]==N)
```

```
N
--
24
```

AUTOMATA PROGRAMMING PARADIGM

1.

```
from automata.fa.dfa import DFA
# lang accepts even number of '0's and even number of '1's
dfa = DFA(
    states={'q0', 'q1', 'q2','q3'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q2', '1': 'q1'},
        'q1': {'1': 'q0', '0': 'q3'},
        'q2': {'0': 'q0', '1': 'q3'},
        'q3': {'0': 'q1', '1': 'q2'}
    },
    initial_state='q0',
    final_states={'q0'}
)
if dfa.accepts_input('101101'):
    print('accepted')
else:
    print('rejected')
```

```
PS C:\Users\91800\OneDrive\Desktop\Week 7> python -u "c:\Users\91800\OneDrive\Desktop\Week 7\automata4.py"
accepted
PS C:\Users\91800\OneDrive\Desktop\Week 7>
```

2.

```
from automata.fa.dfa import DFA
# lang accepts only '101'
dfa = DFA(
    states={'q0', 'q1', 'q2','q3','d'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'd', '1': 'q1'},
        'q1': {'1': 'd', '0': 'q2'},
        'q2': {'0': 'd', '1': 'q3'},
        'q3': {'0': 'd', '1': 'd'},
        'd': {'0': 'q0', '1': 'q3'}
    },
    initial_state='q0',
    final_states={'q3'}
)
if dfa.accepts_input('101'):
    print('accepted')
else:
    print('rejected')
```

```
PS C:\Users\91800\OneDrive\Desktop\Week 7> python -u "c:\Users\91800\OneDrive\Desktop\Week 7\automata4.py"
accepted
PS C:\Users\91800\OneDrive\Desktop\Week 7>
```

3.

```python
from automata.fa.dfa import DFA
# lang accepts consecutive three 1's
dfa = DFA(
    states={'q0', 'q1', 'q2','q3'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'1': 'q2', '0': 'q0'},
        'q2': {'0': 'q0', '1': 'q3'},
        'q3': {'0': 'q3', '1': 'q3'}
    },
    initial_state='q0',
    final_states={'q3'}
)
if dfa.accepts_input('01110'):
    print('accepted')
else:
    print('rejected')
```

```
PS C:\Users\91800\OneDrive\Desktop\Week 7> python -u "c:\Users\91800\OneDrive\Desktop\Week 7\automata2.py"
accepted
PS C:\Users\91800\OneDrive\Desktop\Week 7>
```