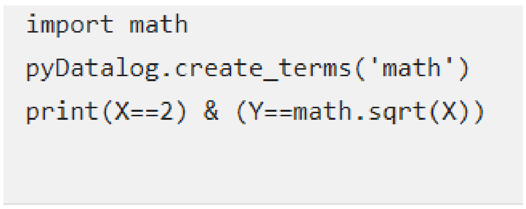
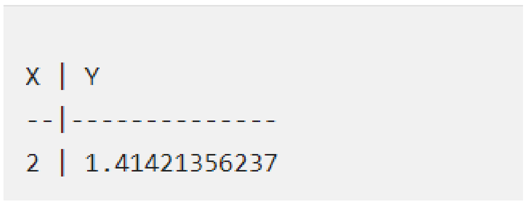
APP CT-3 EXAM B1 – SET A

11)

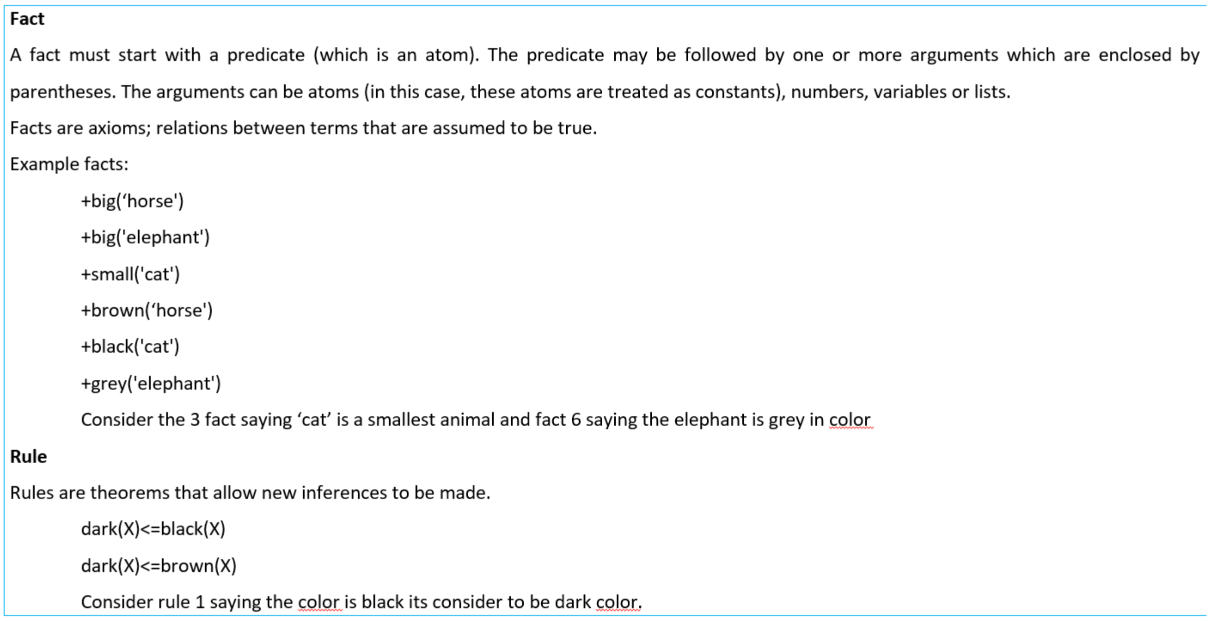


```
import math
pyDatalog.create_terms('math')
print(X==2) & (Y==math.sqrt(X))
```



```
X | Y
--|--------------
2 | 1.41421356237
```

12)



Fact

A fact must start with a predicate (which is an atom). The predicate may be followed by one or more arguments which are enclosed by parentheses. The arguments can be atoms (in this case, these atoms are treated as constants), numbers, variables or lists.

Facts are axioms; relations between terms that are assumed to be true.

Example facts:

```
+big(‘horse')
+big('elephant')
+small('cat')
+brown(‘horse')
+black('cat')
+grey('elephant')
```

Consider the 3 fact saying ‘cat’ is a smallest animal and fact 6 saying the elephant is grey in color

Rule

Rules are theorems that allow new inferences to be made.

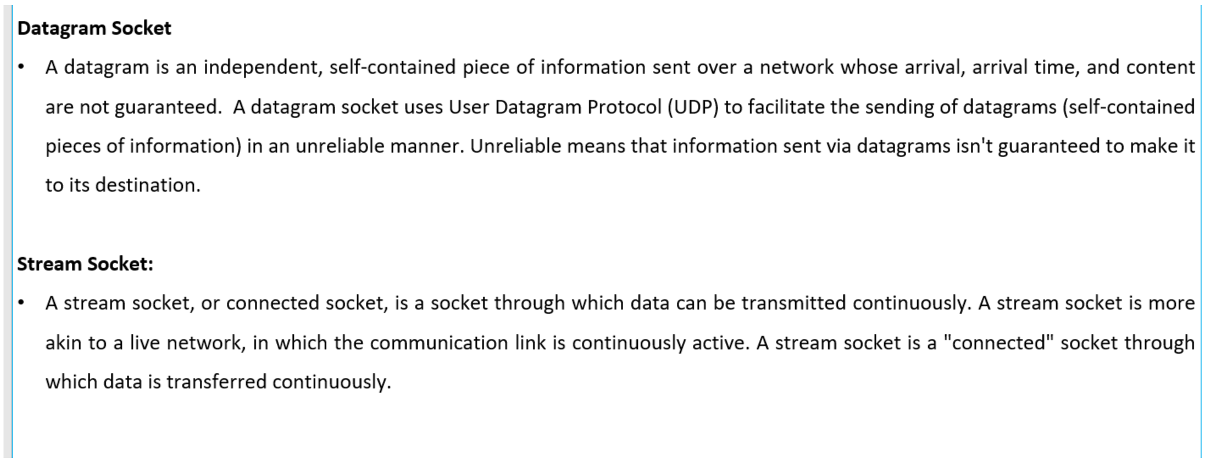
```
dark(X)<=black(X)
dark(X)<=brown(X)
```

Consider rule 1 saying the color is black its consider to be dark color.

Advantages

- The system solves the problem, so the programming steps themselves are kept to a minimum;
- Proving the validity of a given program is simple.

13)



Datagram Socket

- A datagram is an independent, self-contained piece of information sent over a network whose arrival, arrival time, and content are not guaranteed. A datagram socket uses User Datagram Protocol (UDP) to facilitate the sending of datagrams (self-contained pieces of information) in an unreliable manner. Unreliable means that information sent via datagrams isn't guaranteed to make it to its destination.

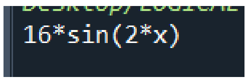
Stream Socket:

- A stream socket, or connected socket, is a socket through which data can be transmitted continuously. A stream socket is more akin to a live network, in which the communication link is continuously active. A stream socket is a "connected" socket through which data is transferred continuously.

14)

a)

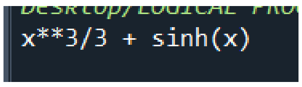
```
import sympy as sym
x, y = sym.symbols('x,y')
print(sym.diff(-8*sym.cos(2*x),x))
```



```
16*sin(2*x)
```

b)

```
import sympy as sym
x= sym.symbols('x')
print(sym.integrate(x**2+sym.cosh(x),x))
```

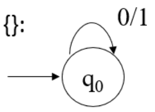


```
x**3/3 + sinh(x)
```

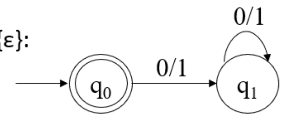
15)

1. Let Σ = {0, 1}. Give DFAs for {}, {ε}, Σ^*, and Σ^+.

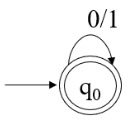
For {}:



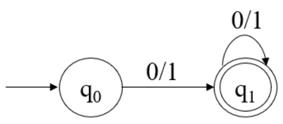
For {ε}:



For Σ^*:

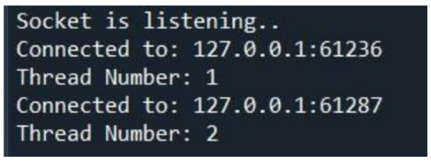


For Σ^+:

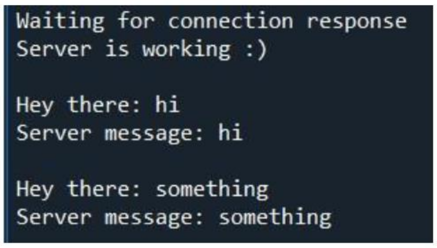


16)

```
# echo-server.py
import socket
import os
from _thread import *
ServerSideSocket = socket.socket()
host = '127.0.0.1'
port = 2004
ThreadCount = 0
try:
    ServerSideSocket.bind((host, port))
except socket.error as e:
    print(str(e))
print('Socket is listening..')
ServerSideSocket.listen(5)
def multi_threaded_client(connection):
    connection.send(str.encode('Server is working :)'))
    while True:
        data = connection.recv(2048)
        response = 'Server message: ' + data.decode('utf-8')
        if not data:
            break
        connection.sendall(str.encode(response))
    connection.close()
while True:
    Client, address = ServerSideSocket.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(multi_threaded_client, (Client, ))
    ThreadCount += 1
    print('Thread Number: ' + str(ThreadCount))
ServerSideSocket.close()

#echo client.py
import socket
ClientMultiSocket = socket.socket()
host = '127.0.0.1'
port = 2004
print('Waiting for connection response')
try:
    ClientMultiSocket.connect((host, port))
except socket.error as e:
    print(str(e))
res = ClientMultiSocket.recv(1024)
print(str(res,'utf-8'))
while True:
    Input = input('Hey there: ')
    ClientMultiSocket.send(str.encode(Input))
    res = ClientMultiSocket.recv(1024)
    print(res.decode('utf-8'))
ClientMultiSocket.close()
```



```
Socket is listening..
Connected to: 127.0.0.1:61236
Thread Number: 1
Connected to: 127.0.0.1:61287
Thread Number: 2
```



```
Waiting for connection response
Server is working :)

Hey there: hi
Server message: hi

Hey there: something
Server message: something
```

17)

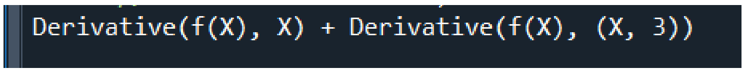
```
from pyDatalog import pyDatalog
pyDatalog.create_terms('X,Y,Z,bear,elephant,cat,small,big,brown,black,gray,dark')
+big('elephant')
+big('bear')
+small('cat')
+black('cat')
+brown('bear')
+gray('elephant')
dark(X)<=black(X)
dark(X)<=brown(X)
print(big(X) & dark(X))
print(big(X) & gray(X))
print(small(X) & black(X))
print(black(X))
print(brown(X))
```

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

18)

i)

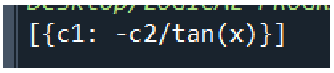
```
import sympy as sym
X=sym.symbols('X')
f=sym.Function('f')
print(sym.diff(f(X)+sym.diff(sym.diff(f(X),X),X),X))
```



```
Derivative(f(X), X) + Derivative(f(X), (X, 3))
```

ii)

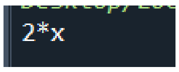
```
import sympy as sym
x,c1,c2=sym.symbols('x,c1,c2')
print(sym.solve(c1*sym.sin(x)+c2*sym.cos(x)))
```



```
[{c1: -c2/tan(x)}]
```

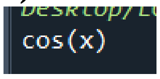
iii)

```
import sympy as sym
x,y=sym.symbols('x,y')
print(((x+y)+(x-y)))
```



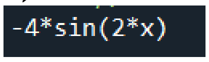
```
2*x
```

iv)
a)



```
cos(x)
```

b)

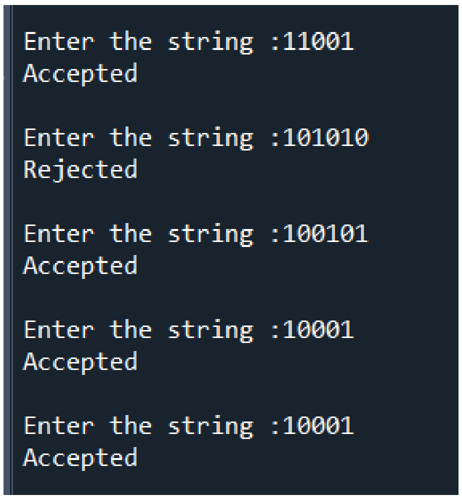


```
-4*sin(2*x)
```

19)

i)

```
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(1,6):
    num = input("Enter the string :")
    if(dfa.accepts_input(num)):
        print("Accepted")
    else:
        print("Rejected")
```

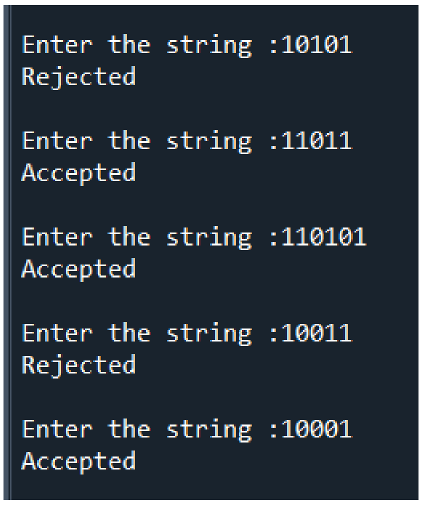


```
Enter the string :11001
Accepted

Enter the string :101010
Rejected

Enter the string :100101
Accepted

Enter the string :10001
Accepted

Enter the string :10001
Accepted
```

ii)

```
from automata.fa.dfa import DFA
dfa = DFA(
states={'q0', 'q1'}, input_symbols={'0', '1'},
transitions={
'q0': {'0': 'q0', '1': 'q1'},
'q1': {'0': 'q1', '1': 'q0'}
},
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 :10101
Rejected

Enter the string :11011
Accepted

Enter the string :110101
Accepted

Enter the string :10011
Rejected

Enter the string :10001
Accepted
```