\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: Shravan Ajit Chinchkar Std: SE(B) Roll No.:140

Practical: 2 Sub: Data Structure and Algorithm

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Program: To create ADT that implement the “set” concept.

a) Add (new Element)-Place a value into the set, b) Remove (element) Remove the value, c) Contains (element) Return true if element is in collection, d) size() Return number of values in collection Iterator() Return an iterator used of loop over collection, e) Intersection of two sets, f) Union of two sets, g) Difference between two sets, h)subset

**Source Code:**

#set operation

def setCreate():

l1=eval(input("Enter values in set:"))

return(set(l1))

def setUpdate(set1,data): #can be used for union of set

s1=set(data) #single data so use {} else set(list)

s1.update(set1)

return(s1)

def setRemove(set1,data):

if(setContains(set1,data)):

set1.remove(data)

return(set1)

def setContains(set1,data):

s1={data}

if(s1.issubset(set1)):

return(True)

else:

return(False)

def setSize(s1):

return(len(s1))

def union(s1,s2):

return(s1.union(s2))

def diff(s1,s2):

s1-=s2

return(s1)

def intersect(s1,s2):

s1 &=s2

return(s1)

def symmetric\_diff(s1,s2):

s1 ^=s2

return(s1)

#driver

s1=setCreate()

s1=setUpdate(s1,[1,2,6]) # Same as Union

print(s1)

s1=setRemove(s1,6)

print(s1)

print(setContains(s1,5))

print(setContains(s1,6))

s1=setUpdate({1,2,3,4,5},{4,5,6}) #alternative to union

print(s1)

print(union({1,2,3,4,5},{4,5,6}))

print(diff({1,2,3,4,5},{4,5,6}))

print(intersect({1,2,3,4,5},{4,5,6}))

print(symmetric\_diff({1,2,3,4,5},{4,5,6}))

print(setSize({1,2,3,4,5}))

**Output:**

Enter values in set:1,

{1, 2, 6}

{1, 2}

False

False

{1, 2, 3, 4, 5, 6}

{1, 2, 3, 4, 5, 6}

{1, 2, 3}

{4, 5}

{1, 2, 3, 6}

5