# Roll no : COA244

# Assignment no : 2

class SetADT:

def init (self):

self.elements = {} def add(self, element):

self.elements[element] = None def remove(self, element):

if element in self.elements: del self.elements[element]

else:

raise KeyError(f"Element {element} not found in the set") def contains(self, element):

return element in self.elements def size(self):

return len(self.elements) def iterator(self):

return iter(self.elements)

def intersection(self, other\_set):

result = SetADT()

for element in self.elements:

if other\_set.contains(element): result.add(element)

return result

def union(self, other\_set):

result = SetADT()

for element in self.elements: result.add(element)

for element in other\_set.elements: result.add(element)

return result

def difference(self, other\_set):

result = SetADT()

for element in self.elements:

if not other\_set.contains(element): result.add(element)

return result

def subset(self, other\_set):

for element in self.elements:

if not other\_set.contains(element): return False

return True

if name == " main ": set1 = SetADT() set1.add(1) set1.add(2) set1.add(3)

set2 = SetADT() set2.add(2) set2.add(3) set2.add(4)

print("Set 1 contains 2:", set1.contains(2)) print("Set 1 size:", set1.size()) set1.remove(2)

print("Set 1 after removing 2:", [x for x in set1.iterator()]) union\_set = set1.union(set2)

print("Union of Set 1 and Set 2:", [x for x in union\_set.iterator()])

intersection\_set = set1.intersection(set2) print("Intersection of Set 1 and Set 2:", [x for x in

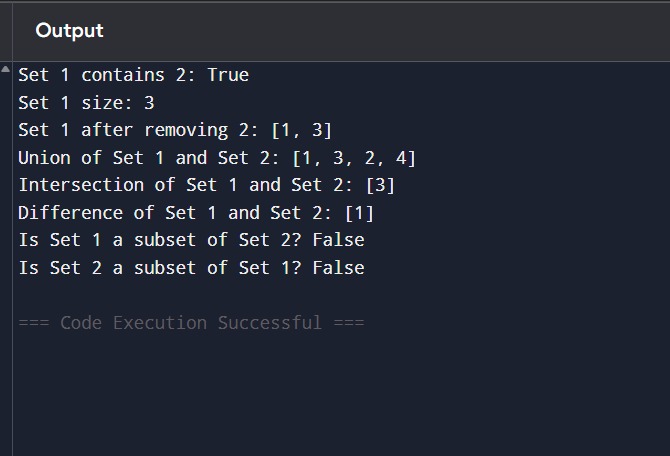
intersection\_set.iterator()])

difference\_set = set1.difference(set2) print("Difference of Set 1 and Set 2:", [x for x in

difference\_set.iterator()])

print("Is Set 1 a subset of Set 2?", set1.subset(set2)) print("Is Set 2 a subset of Set 1?", set2.subset(set1))

OUTPUT :



# 