```
/usr/local/bin/python3.12 /Users/pengyenjia/Desktop/運算思維與程式設計/makeUp_Submission_py/5_27/課堂練習/11227130_資訊二甲_11227130_彭妍嘉 5_27.py
First circle object
First circle's area: 78.53981633974483
First circle's perimeter: 31.41592653589793
Second circle object
Second circle's area: 314.1592653589793
Second circle's perimeter: 62.83185307179586
Student name:  張曉明
student height:  170
student weight:  70
student BMI:  24.221453287197235
student name:  阿敏
student height:  169
student weight:  66
student BMI:  23.1084345786212
Stack: [1, 2]
Queue: [2, 3]
DisjointSet: { 1 ,2 ,3 },{ 4 ,5 },{ 6 ,7 ,8 },
Find(1) = 1
Find(2) = 1
Find(3) = 1
Find(4) = 4
Find(5) = 4
Find(6) = 6
Find(7) = 6
Find(8) = 6
```

```python
# 13-1
import math

class Circle:  2 usages  new *
    def __init__(self,radius):  new *
        self.radius = radius
    def getArea(self):  2 usages  new *
        return self.radius * self.radius * math.pi
    def getPerimeter(self):  2 usages  new *
        return self.radius * 2 * math.pi

circle1 = Circle(5)
print("First circle object")
print("First circle's area:", circle1.getArea())
print("First circle's perimeter:", circle1.getPerimeter())
circle2 = Circle(10)
print("Second circle object")
print("Second circle's area:", circle2.getArea())
print("Second circle's perimeter:", circle2.getPerimeter())
# 13-2
class Student:  2 usages  new *
    def __init__(self,name,height, weight):  new *
        self.name = name
        self.height = height
        self.weight = weight

    def getName(self):  2 usages  new *
        return self.name
    def getHeight(self):  2 usages  new *
        return self.height
    def getWeight(self):  2 usages  new *
        return self.weight
    def getBMI(self):  2 usages  new *
        return self.weight/(self.height/100)**2
```

```python
student1 = Student( name: "張曉明",  height: 170,  weight: 70)
print("Student name: ", student1.getName())
print("student height: ", student1.getHeight())
print("student weight: ", student1.getWeight())
print("student BMI: ", student1.getBMI())
student2 = Student( name: "阿敏",  height: 169,  weight: 66)
print("student name: ", student2.getName())
print("student height: ", student2.getHeight())
print("student weight: ", student2.getWeight())
print("student BMI: ", student2.getBMI())

#13-3
class Stack:  1 usage  new *
    def __init__(self):  new *
        self.s = []
    def isEmpty(self):  1 usage  new *
        return self.s == []
    def Push(self, key):  3 usages  new *
        self.s.append(key)
    def Pop(self):  1 usage  new *
        if self.isEmpty():
            print("Stack is empty (Underflow)")
            return None
        else:
            return self.s.pop()
    def Display(self):  1 usage  new *
        print("Stack: ", end = "")
        print(self.s)
s = Stack()
s.Push(1)
s.Push(2)
s.Push(3)
s.Pop()
s.Display()
```

```
62          print(self.s)
63  s = Stack()
64  s.Push(1)
65  s.Push(2)
66  s.Push(3)
67  s.Pop()
68  s.Display()
69
70  #13-4
71  class Queue:  1 usage  new *
72      def __init__(self):  new *
73          self.Q = []
74      def isEmpty(self):  1 usage  new *
75          return self.Q == []
76      def enQueue(self, item):  3 usages  new *
77          self.Q.append(item)
78      def deQueue(self):  1 usage  new *
79          if self.isEmpty():
80              print("Underflow")
81              return None
82          else:
83              return self.Q.pop(0)
84      def Display(self):  1 usage  new *
85          print("Queue: ", end = "")
86          print(self.Q)
87  Q = Queue()
88  Q.enQueue(1)
89  Q.enQueue(2)
90  Q.enQueue(3)
91  Q.deQueue()
92  Q.Display()
93
94  #13-5
95  class DisjointSet:  1 usage  new *
```

```
92      Q.Display()
93
94      #13-5
95      class DisjointSet:  1 usage  new *
96          def __init__(self, n):  new *
97              self.set = [i for i in range(n + 1)]
98              self.n = n
99          def Find(self, key):  11 usages  new *
100             while self.set[key] != key:
101                 key = self.set[key]
102             return key
103         def Union(self, a, b):  5 usages  new *
104             if self.Find(a) < self.Find(b):
105                 for i in range(self.n + 1):
106                     if self.Find(i) == self.Find(b):
107                         self.set[i] = self.Find(a)
108             else:
109                 for i in range(self.n + 1):
110                     if self.Find(i) == self.Find(a):
111                         self.set[i] = self.Find(b)
112         def Display(self):  1 usage  new *
113             print("DisjointSet: ", end = "")
114             for i in range(1, self.n + 1):
115                 if self.Find(i) == i:
116                     print("{ ", end = "")
117                     print(i, end = " ")
118                     for j in range(i + 1, self.n + 1):
119                         if self.Find(j) == i:
120                             print(",", end = "")
121                             print(j, end = " ")
122                     print("},", end = "")
123             print()
124
125     n = 8
```

```
95    class DisjointSet:  1 usage  new *
103       def Union(self, a, b):  5 usages  new *
110                   if self.Find(i) == self.Find(a):
111                       self.set[i] = self.Find(b)
112       def Display(self):  1 usage  new *
113           print("DisjointSet: ", end = "")
114           for i in range(1, self.n + 1):
115               if self.Find(i) == i:
116                   print("{ ", end = "")
117                   print(i, end = " ")
118                   for j in range(i + 1, self.n + 1):
119                       if self.Find(j) == i:
120                           print(",", end = "")
121                           print(j, end = " ")
122                   print("},", end = "")
123           print()
124
125   n = 8
126   S = DisjointSet(n)
127   S.Union( a: 1,  b: 2)
128   S.Union( a: 1,  b: 3)
129   S.Union( a: 4,  b: 5)
130   S.Union( a: 6,  b: 7)
131   S.Union( a: 7,  b: 8)
132   S.Display()
133   for i in range(1, n + 1):
134       print("Find(%d) = %d" % (i, S.Find(i)))
135
136
```