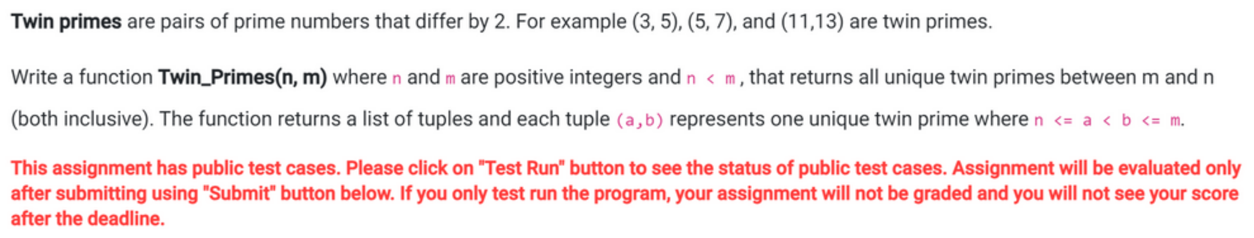
PPA-1



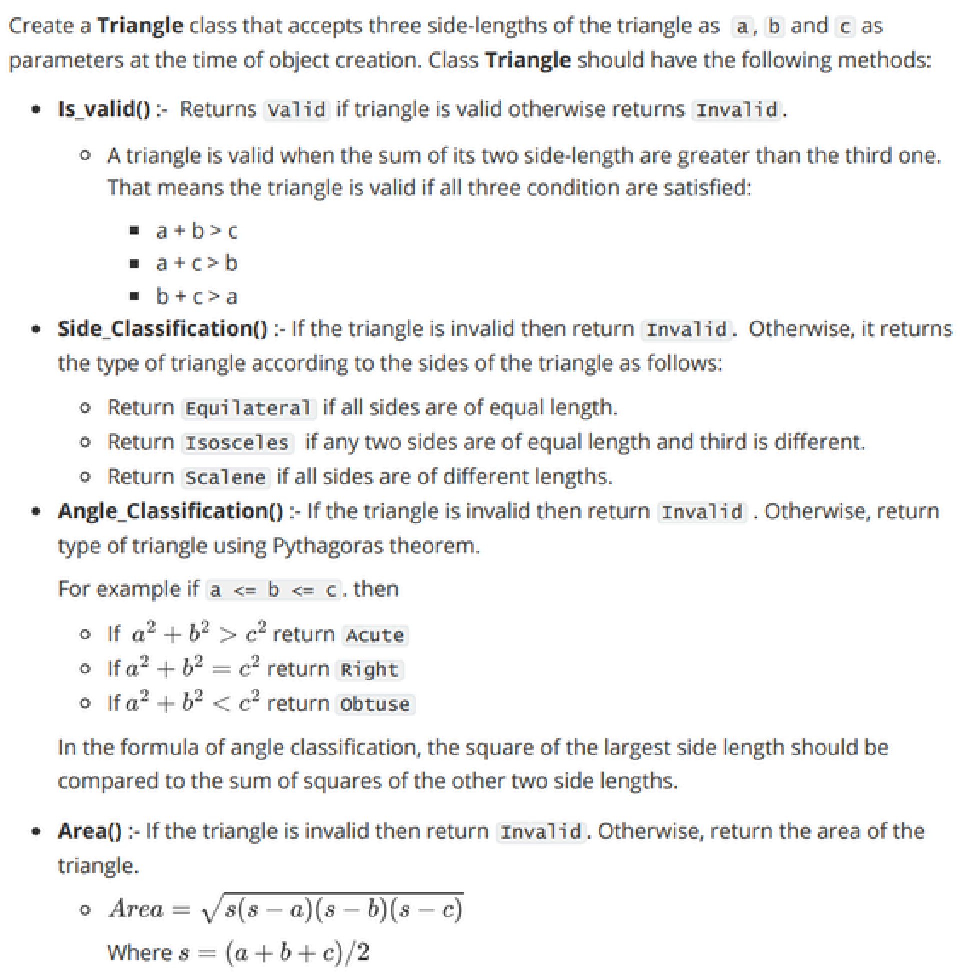
Twin primes are pairs of prime numbers that differ by 2. For example (3, 5), (5, 7), and (11,13) are twin primes.

Write a function **Twin_Primes(n, m)** where `n` and `m` are positive integers and `n < m`, that returns all unique twin primes between m and n (both inclusive). The function returns a list of tuples and each tuple `(a,b)` represents one unique twin prime where `n <= a < b <= m`.

This assignment has public test cases. Please click on "Test Run" button to see the status of public test cases. Assignment will be evaluated only after submitting using "Submit" button below. If you only test run the program, your assignment will not be graded and you will not see your score after the deadline.

```
def factors(n):
    factor_list = []
    for i in range(1,(n+1)):
        if n % i == 0:
            factor_list.append(i)
    return factor_list

def prime1(temp1, n):
    if temp1 == [1, n]:
        return True
    else:
        return False

def Twin_Primes(n,m):
    list1 = []
    for i in range(n,m+1):
        j = i +2
        if prime1(factors(i),i) and prime1(factors(j),j):
            list1.append((i,j))
    return list1



n=int(input())
m=int(input())
print(sorted(Twin_Primes(n, m)))
```

PPA-2



Create a **Triangle** class that accepts three side-lengths of the triangle as `a`, `b` and `c` as parameters at the time of object creation. Class **Triangle** should have the following methods:

- **Is_valid()** :- Returns `Valid` if triangle is valid otherwise returns `Invalid`.
  - A triangle is valid when the sum of its two side-length are greater than the third one. That means the triangle is valid if all three condition are satisfied:
    - a + b > c
    - a + c > b
    - b + c > a
- **Side_Classification()** :- If the triangle is invalid then return `Invalid`. Otherwise, it returns the type of triangle according to the sides of the triangle as follows:
  - Return `Equilateral` if all sides are of equal length.
  - Return `Isosceles` if any two sides are of equal length and third is different.
  - Return `Scalene` if all sides are of different lengths.
- **Angle_Classification()** :- If the triangle is invalid then return `Invalid`. Otherwise, return type of triangle using Pythagoras theorem.

  For example if `a <= b <= c`, then

  - If $a^2 + b^2 > c^2$ return `Acute`
  - If $a^2 + b^2 = c^2$ return `Right`
  - If $a^2 + b^2 < c^2$ return `Obtuse`

  In the formula of angle classification, the square of the largest side length should be compared to the sum of squares of the other two side lengths.
- **Area()** :- If the triangle is invalid then return `Invalid`. Otherwise, return the area of the triangle.
  - $Area = \sqrt{s(s-a)(s-b)(s-c)}$

    Where $s = (a+b+c)/2$

```
class Triangle:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def Is_valid(self):
        if (self.a + self.b > self.c) and (self.a + self.c > self.b) and (self.b + self.c >
self.a):
            return 'Valid'
        else:
            return 'Invalid'
    def Side_Classification(self):
        if self.Is_valid() == 'Valid':
            if self.a == self.b and self.b == self.c:
                return 'Equilateral'
            elif (self.a == self.b) or (self.a == self.c) or (self.b == self.c):
                return 'Isosceles'
            else:
                return 'Scalene'
        else:
            return 'Invalid'

    def Angle_Classification(self):
        if self.Is_valid() == 'Valid':
            if (self.a * self.a) + (self.b * self.b) > (self.c * self.c):
                return 'Acute'
            elif (self.a * self.a) + (self.b * self.b) == (self.c * self.c):
                return 'Right'
            else:
                return 'Obtuse'
        else:
            return 'Invalid'

    def Area(self):
        import math
        if self.Is_valid() == 'Valid':
            s = (self.a + self.b + self.c)/2
            return math.sqrt(s*(s-self.a)*(s-self.b)*(s-self.c))
        else:
            return 'Invalid'

a=int(input())
b=int(input())
c=int(input())
T=Triangle(a,b,c)
print(T.Is_valid())
print(T.Side_Classification())
print(T.Angle_Classification())
print(T.Area())
```