CSE470: Software Engineering

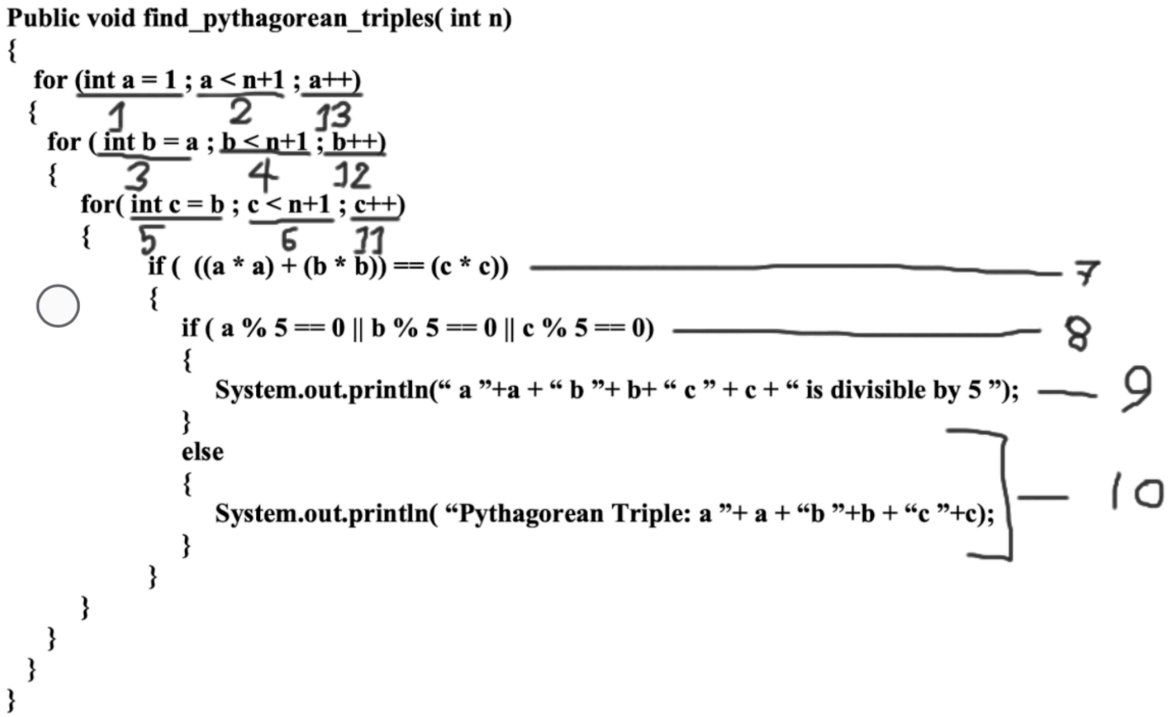
Assignment 3

Name: Umme Abira Azmary

ID: 20101539

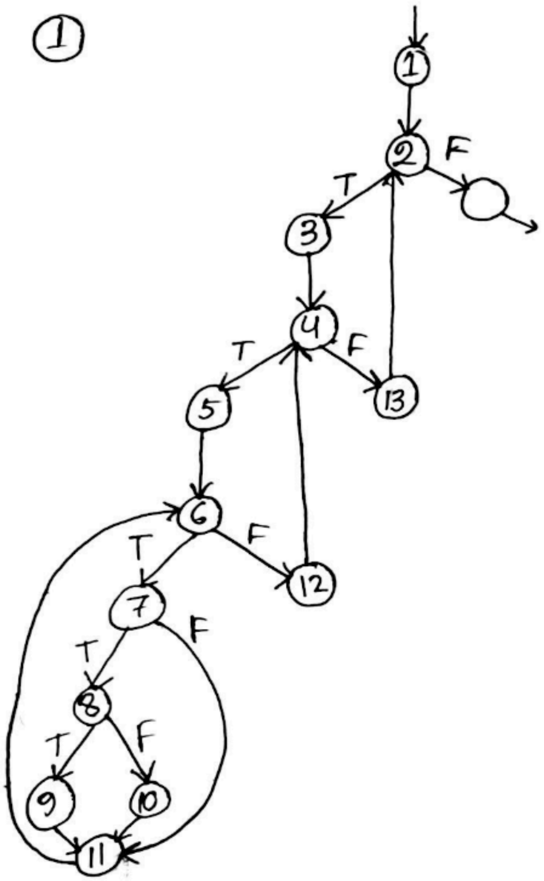
Section: 04

Answer to the Question no - 01:



```
Public void find_pythagorean_triples( int n)
{
  for (int a = 1 ; a < n+1 ; a++)
  {      1          2      13
   for ( int b = a ; b < n+1 ; b++)
   {      3          4       12
     for( int c = b ; c < n+1 ; c++)
     {     5           6      11
        if (  ((a * a) + (b * b)) == (c * c)) ——————————— 7
  ○     {
          if ( a % 5 == 0 || b % 5 == 0 || c % 5 == 0) ——————— 8
          {
             System.out.println(“ a ”+a + “ b ”+ b+ “ c ” + c + “ is divisible by 5 ”); — 9
          }
          else                                                        ⌉
          {                                                           |— 10
             System.out.println( “Pythagorean Triple: a ”+ a + “b ”+b + “c ”+c); |
          }                                                           ⌋
        }
     }
   }
  }
}
```

①



②

$M = R + 1$
$= 5 + 1$
$= 6$

$M = P + 1$
$= 5 + 1$
$= 6$

$M = E - N + 2P$
$= 18 - 14 + 2 \times 1$
$= 4 + 2$
$= 6$.

③

→ 1 → 2 → ○ →

→ 1 → 2 → 3 → 4 → 13 → 2 → ○ →

→ 1 → 2 → 3 → 4 → 5 → 6 → 12 → 4 → 13 → 2 → ○ →

→ 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 11 → 6 → 12 → 4 → 13
→ 2 → ○ →

→ 1 → 2 → 3 → 4 → 5 → 6 → 7 → 11 → 6 → 12 → 4 → 13 → 2 → ○ →

→(1)→(2)→(3)→(4)→(5)→(6)→(7)→(8)→(10)→(11)→(6)→(12)→(4)→(13)→(2)→( )→

So, we have total 6 Independent paths.

Now, let's use the fallowing ~~path~~ path to showcase a test case:

→(1)→(2)→(3)→(4)→(5)→(6)→(12)→(4)→(13)→(2)→( )→

<u>Condition</u>

<u>Node 1;</u> $a=1$, Let's assume $n=1$; <u>Node 2:</u> $a<n+1$.
$1<1+1$.
$1<2$

<u>Node 3:</u> $b=1$, <u>Node 4:</u> $1<1+1$
$1<2$

<u>Node 5:</u> $c=1$, <u>Node 6:</u> $c<n+1$
$1<1+1$
$1<2$

<u>Node 7:</u> Not satisfied, so go back to <u>Node 12</u>: $b=1+1$
$b=2$

Now, <u>Node 4:</u> $b<n+1$.
$2<\cancel{2}1+1$
$2<2$ → Not satisfied

So, go back to <u>Node 13:</u> $a=1+1$
$a=2$

<u>Node 2:</u> $a<n+1$
$2<1+1$
$2<2$ → Not satisfied

<u>Node Logical:</u> Exit through Logical Node.

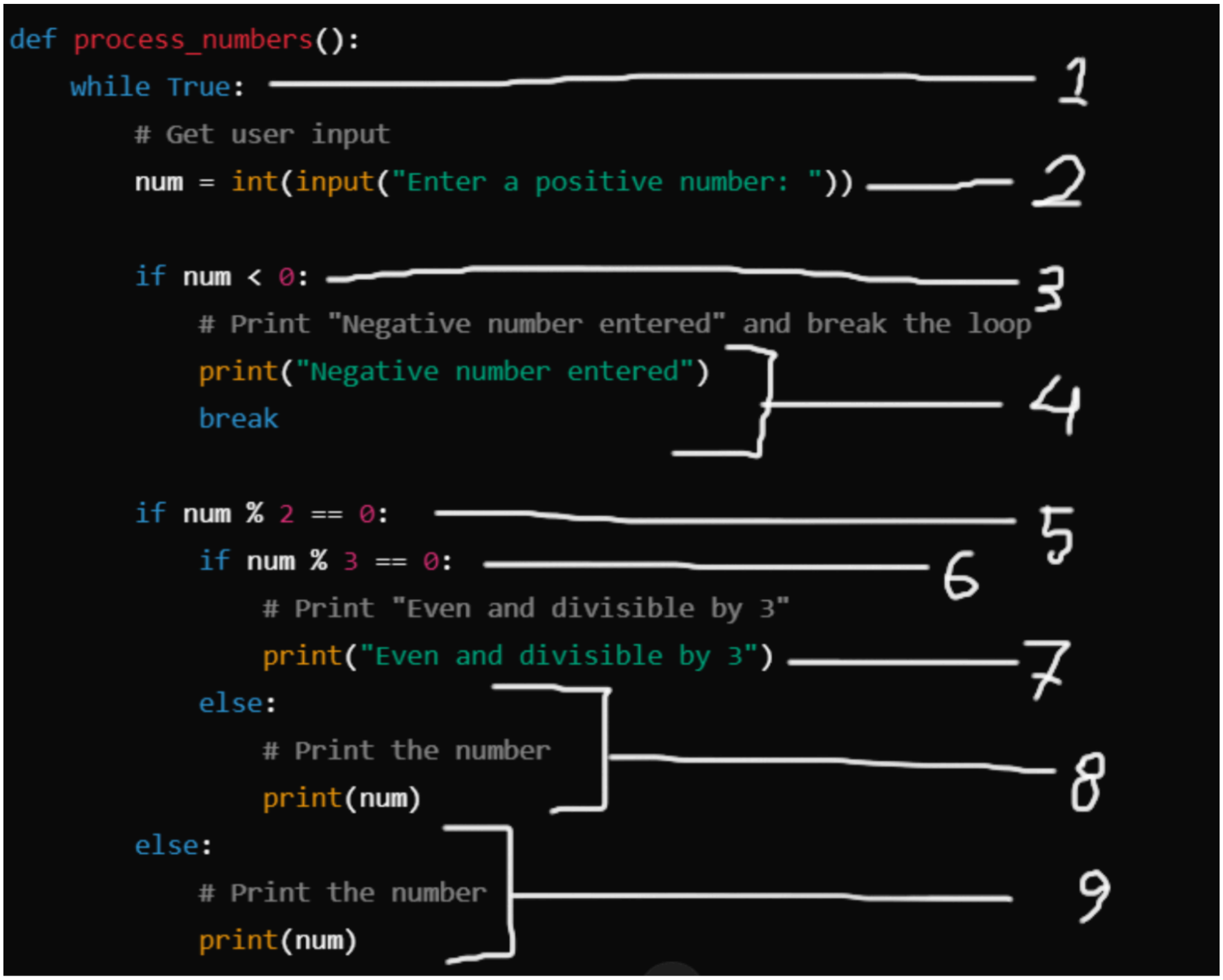
④ Here, no. of Independent path = 6 .

weknow, if no. of Independent path <= M, our path-based testing is done correctly.

$$6 <= 6 .$$

so, we understand our path-based testing is done correctly.

Answer to the question no -02:

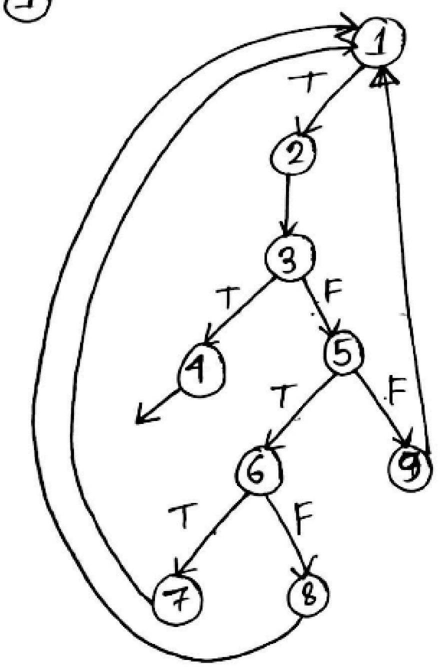


```python
def process_numbers():
    while True:
        # Get user input
        num = int(input("Enter a positive number: "))

        if num < 0:
            # Print "Negative number entered" and break the loop
            print("Negative number entered")
            break

        if num % 2 == 0:
            if num % 3 == 0:
                # Print "Even and divisible by 3"
                print("Even and divisible by 3")
            else:
                # Print the number
                print(num)
        else:
            # Print the number
            print(num)
```

①



② $M = R+1.$

$= 9+1.$

$= 4$

$M = P+1.$

$= 3+1.$

$= 4$

$M = E - N + 2P$

$= (11-9) + 2 \times 1$

$= 2+2$

$= 4.$

③ <u>Independent Path:</u>

→ ① → ② → ③ → ④ →

→ ① → ② → ③ → ⑤ → ⑥ → ⑦ → ① → ② → ③ → ④ →

→ ① → ② → ③ → ⑤ → ⑨ → ① → ② → ③ → ④ →

→ ① → ② → ③ → ⑤ → ⑥ → ⑧ → ① → ② → ③ → ④ → .

Here, we have total 4 independent paths.

Now, let's use the following path to showcase a test case:

→①→②→③→⑤→⑥→⑦→①→②→③→④→

Node 1: While True

Node 2: num = 12

Node 3: Not satisfied.

Node 5: 12 % 2 == 0 :

Node 6: 12 % 3 == 0 ;

Node 7: print (" Even and divisible by 3")

Node 1: While True continue

Node 2: num = -1

Node 3: num < 0 :
    -1 < 1 (satisfied)

Node 4: print(" Negative Number entered")
    break.

④ We know,

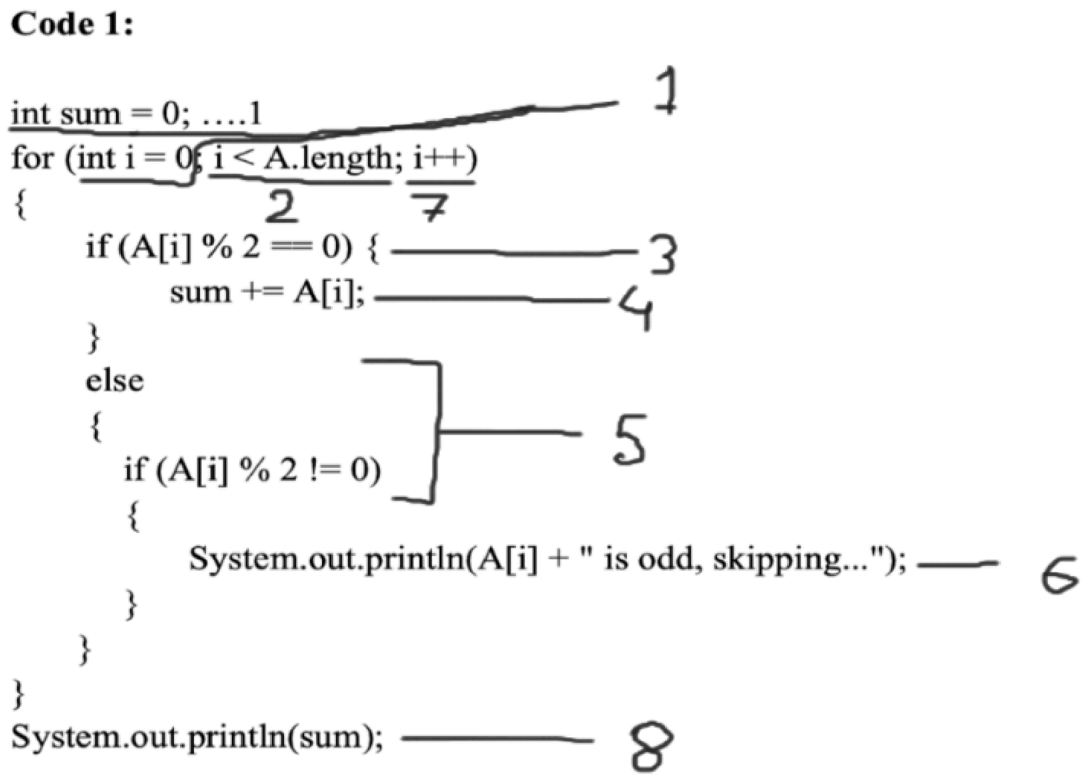
No. of Independent Path <= M.

4 <= M.

4 < =4.

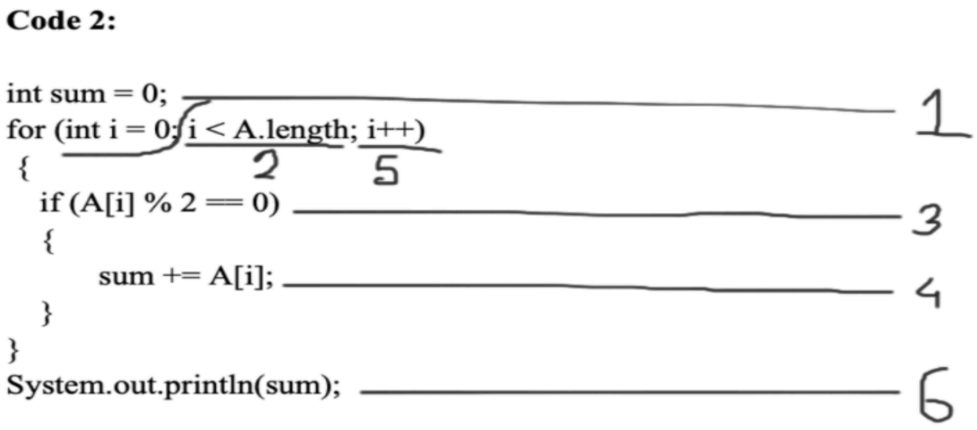
So, we can say, our path-based testing is done correctly.

Answer to the question no - 03:



Code 1:

```
int sum = 0; ….1
for (int i = 0; i < A.length; i++)
{
      if (A[i] % 2 == 0) {
            sum += A[i];
      }
      else
      {
         if (A[i] % 2 != 0)
         {
              System.out.println(A[i] + " is odd, skipping...");
         }
      }
}
System.out.println(sum);
```

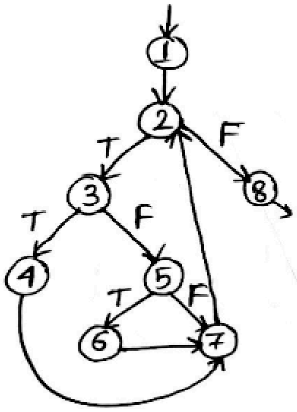


Code 2:

```
int sum = 0;
for (int i = 0; i < A.length; i++)
 {
   if (A[i] % 2 == 0)
   {
        sum += A[i];
   }
}
System.out.println(sum);
```

Answer to the ques. no-03:

① Code1 :

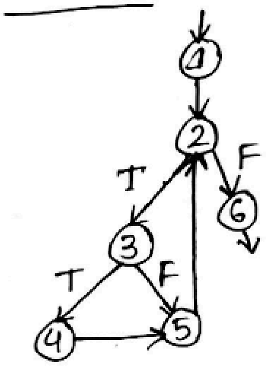


Cyclometric complexity, $M = R+1$
$= 3+1$
$= 4$.

$M = P+1$
$= 3+1$
$= 4$

$M = E - N + 2P$
$= (10-8) + 2\times1$
$= 4$.

Code 2 :



Cyclometric complexity, $M = R+1$.
$= 2+1$
$= 3$

$M = P+1$.
$= 2+1$
$= 3$

$M = E - N + 2P$.
$= (7-6) + 2\times1$.
$= 3$

Here, Code 2 has better Cyclometric Complexity.

(2) Code 1,

Independent Paths:

→ 1 → 2 → 8 →

→ 1 → 2 → 3 → 4 → 7 → 2 → 8 →

→ 1 → 2 → 3 → 5 → 6 → 7 → 2 → 8 →

→ 1 → 2 → 3 → 5 → 7 → 2 → 8 → .

Code 2,

Independent Paths:

→ 1 → 2 → 6 →

→ 1 → 2 → 3 → 4 → 5 → 2 → 6 →

→ 1 → 2 → 3 → 5 → 2 → 6 →

For Code 1, → 1 → 2 → 8 →

Node 1 : int sum = 0
int i = 0

Node 2 : A = [ ]
i < A.length.
0 < 0 → Not satisfied

Node 8 : print (sum)
↳ sum = 0.

→ 1 → 2 → 3 → 4 → 7 → 2 → 8 →

Node 1 : int sum = 0
int i = 0

Node 2 : A = [2,4] , i < A.length
0 < 2

Node 3 : A[0] % 2 == 0
2 % 2 == 0

Node 4 : sum = 0+2 ⇒ sum = 2

Node 7 : i = 0+1 = 1.

Node 2 : 1 < 2.

Node 3 : A[1] ⇒ 4 % 2 == 0 .

Node 4 : sum = 2+4 ⇒ sum = 6 .

Node 7 : i = 1+1 = 2

Node 2 : 2<8 → Not satisfied.

Node 8 : Print (sum = 8) .

→①→②→③→⑤→⑥→⑦→②→⑧→

Node1: sum=0
i=0

Node2: A=[3]

Node 3: A[0] % 2 ==0
3 % 2 ==0 → Not satisfied.

Node 5: A[0] % 2 !=0
3 % 2 !=0 ⇒

Node 6: print(3 is odd, skipping...)

Node 7: i=0+1.

Node 2 = A[1] % 2 ==0 → Not ~~stat~~ satisfied.

Node8: sum=3

→①→②→③→⑤→⑦→②→⑧→

Node 1: sum=0
i=0

Node 2: A=[3, 2]

Node 3: A[0] % 2 ==0
3 % 2 ==0 → Not satisfied.

Node 5: A[0] % 2 !=0
3 % 2 !=0 → This node allows node 6 to print. Which is not a part of this path.

So, there is no such value that can satisfy this path.

Code 2:

→①→②→⑥→

Node1: sum=0
i=0

Node 2: ~~A=[2,4]~~ A=[ ]
i<A.length
0< 0 → Not satisfied

Node 6: sum=0

→①→②→③→④→⑤→②→⑥→

Node1: sum=0
i=0

Node 2: A=[2,4]
i<A.length
0<2

Node 3: A[0] % 2==0
2 % 2==0

Node 4: sum=0+2
sum=2

Node 5: i=0+1 ⇒ i=1

Node 2: 1<2

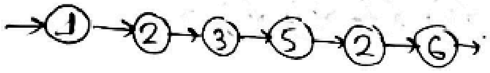
Node 3: A[1] % 2 ==0
4 % 2 ==0

Node 4: sum=2+4 ⇒ sum=6.

Node 5: i=1+1 ⇒ i=2

Node 2: 2<2 → Not satisfied.

Node6: sum= 6.



Node 1: sum = 0
i = 0

Node 2: A = [3]

Node 3: A[0] % 2 == 0
3 % 2 == 0 → Not satisfied.

Node 5: i = 0+1.
~~A[1] % 2 == 0 → Not satisfied~~.

Node 2: A[1] % 2 == 0 → Not satisfied.

Node 6: sum = 0.

Answer to the ques. no:4:

$$SIX = \frac{NMO * DIT}{NMO + NMI + NMA}$$

$NMO = 1$

$NMI = 1$

$NMA = 2$

$DIT = 1.$

So, $$SIX = \frac{1 \cdot 1}{1 + 2 + 1.}$$

$$SIX = \frac{1}{4} \times 100\%$$

$$= 25\%$$