Name: _____ **Roll No:** _____ **Section:**_____

Quiz 1 (IP 2024). **Time: 25 mts.**

- Write your name, roll no, section on top of this page.
- Pls use the space provided after each question to give your answer. You can use the opposite side of the pages for rough work.
- This is an exam, so as per Institute policy, if you are caught using unfair means, you can get an F in the course.

Q1 (2 marks). After studying about loops and functions in his course, Ram writes the following code to test out how nested loops work. Consider the following functions written by him:

```
def nested(n):
    for i in range(n):
          print("Outer")
          for j in range(i):
               print("Inner")
```

If we call the function with the value of n=10 (i.e. for nested(10)) answer the following:

(i) How many times will "Outer" be printed:	10 times
(ii) How many times will "Inner" be printed:	45 times

Explanation:
(i) For each i (0 to 9) " `Outer`" is printed: 10 times.
(ii) For each outer loop i, j's range is (0,i) [(0,0),(0,1),(0,2),...(0,9)] and the inner loop prints `"Inner"` i times : (0+1+2+3+...+9)= (9*10)/2 = 45 times

Q2 (2 marks). Consider the following code:

```
def f(x):
    return x*2

def compose(f, g, x):
    return f(g(x))

def h(f, x, n):
    result = x
    for i in range(n):
        result = compose(f, f, result)
    return result
```

(i). Briefly explain what the function 'h' is computing. Function 'h' is computing x*(4^n)	**(ii) What will be the output for a function call h() on x=2 and n=3:** h(f,x=2,n=3)= 2*(4^3)=128

h(f,x,n) calls compose(f,f,result) → At i=0: compose(f,f,result) = f(f(x)) = f(x*2)= (x*2)*2= x*4; result=x*4 At i =1, compose(f,f,x*4)= x*(4^2) At i=n-1 compose(f,f,x*(4^(n-1))= x*(4^n)	

Q3 (1 mark). Write a comment for this function (a simple sentence) describing what this function is doing.

```
def fn(N, a, b):
    count = 0
    for i in range (N):
        if i%a == 0:
            if i%b == 0:
                count += 1
    return count
```

Answer: For numbers from **0 to N-1**, the function counts **how many numbers are divisible by both a and b**.

Q4 (1 mark). Write a comment for this function (a simple sentence) describing what this function computes.

```
def f():
    num_list = [2, 5, 9, 15, 5]
    p = -1
    result = 0
    for num in num_list:
        result = result + num * p
        p = p * -1
    return result
```

Answer: The function computes result as: sum (numbers in the even position of num_list) - sum (numbers in the odd position of num_list)

(OR) The function computes result as: sum (2nd and 4th numbers of num_list) - sum (1st, 3rd and 5th numbers of num_list)

Explanation: result= 2*(-1) + 5*(+1) +9*(-1) +15*(+1) +5*(-1) = -2+5-9+15-5 =(5+15) -(2+9+5) =4

Q5 (1 mark): The following function to reverse a number has a bug. Write one assert statement that will reveal the bug.

```
def reverse(num):
    ans = 0
    for _ in range(5):
        ans = ans * 10 + (num % 10)
        num //= 10
        if num == 0:
            break
    return ans
```

Assertion statement that will reveal the error:

Assert statement where the code should fail : for numbers with more than 5 digits.

Example:
num = 123456
assert reverse(num) == 654321, "assertion error"

Q6 (3 marks): Consider the following function to determine the 2nd largest number of an integer list.

```
1. def second_largest(lst): # lst is a list of integers e.g. [5,7,3,9]
2.     largest = 0
3.     largest2 = 0
4.     for num in lst:
5.         if num > largest:
6.             largest2 = largest
7.             largest = num
8.     return largest2
```

The function has a bug. Write an assertion statement which will pass (that is the function computes correctly and so the bug is not not revealed), and an assert statement which will fail (and reveal the bug). Suggest minimal modifications to the program to fix the bug (if you want to insert some code, mention the line number after which you will insert).

(i) Assertion which passes:	**(ii) Assertion which shows the bug:**
assert(second_largest([1,2,3,4])==3, "the code works properly, so the line is not printed"	assert(second_largest([5,5,1,2]))==2 ,"test case failed as second largest will be printed 0" #can't handle duplicates

	assert(second_largest([5,1,2,3]))==3 ,"test case failed as second largest will be printed 0" #can't handle when the largest element appears before the second largest element in the list

(iii) Modification to the code:

```
def second_largest(lst):
    largest = 0
    largest2 = 0
    for num in lst:
        if num > largest:
            largest2 = largest
            largest = num
        elif num > largest2 and num!=largest:
            largest2 = num
    if largest2==0: #optional condition where second largest don't exist
        return -1   #when the list has one element or all same elements
    return largest2
```

Note: In Q6 (iii), adding the optional condition (to check for invalid input) alone will not fetch any marks.

Q7. (2 marks) What would be the outcome of the following code snippet?

```
def f(a):
    if a%2 == 0:
        print('calling g()')
        g(a)
    else:
        a += 1
    return

def g(a):
    if a%2 != 0:
        print('calling f()')
        f(a)
    else:
        a -= 1
    return

a = 0
while a < 10:
    f(a)
```

Answer:

calling g()
calling g()
calling g()
calling g()
calling g()

Function g() will be called for even numbers. The *while* loop runs from 0 to 9. So, g() will be called for a=0,2,4,6,8.

`a += 1`	