# C Programming lab Recursion quiz

**Name:** _____

**Email:** _____

**Roll nos:** _____

**Concept:** *verifying code*

1. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a == b)
        return 0;
    else
        return a + sum(a + 1,b);
    }
```

   (a) Yes
   (b) No

2. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a == b) return a;
    else return a + sum(a + 1,b);
    }
```

   (a) No
   (b) Yes

3. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a == b) return b;
    else return b + sum(a + 1,b);
    }
```

   (a) No
   (b) Yes

4. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a == b) return b;
    else return a + sum(a + 1,b);
    }
```

(a) No

(b) Yes

5. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a == b) return 1;
    else return a + sum(a + 1,b);
    }
```

(a) No

(b) Yes

6. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a > b) return 0;
    else return a + sum(a + 1,b);
    }
```

(a) Yes

(b) No

7. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a == b) return a;
    else return b + sum(a,b-1);
    }
```

(a) Yes

(b) No

8. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a == b) return a;
    else return a + sum(a,b-1);
    }
```

(a) No

(b) Yes

9. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
def sum(int a,int b)
    {
    if (a > b) return 0;
    else return b + sum(a,b-1);
    }
```

(a) No

(b) Yes

10. Consider the problem statement: *sum the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int sum(int a,int b)
    {
    if (a > b) return b;
    else return b + sum(a,b-1);
    }
```

(a) No

(b) Yes

11. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a == b)
        return 0;
    else
        return a * prod(a + 1,b);
    }
```

(a) Yes

(b) No

12. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a == b) return a;
    else return a * prod(a + 1,b);
    }
```

(a) No

(b) Yes

13. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a == b) return b;
    else return b * prod(a + 1,b);
    }
```

(a) No

(b) Yes

14. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a == b) return b;
    else return a * prod(a + 1,b);
    }
```

(a) No

(b) Yes

15. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a == b) return 1;
    else return a * prod(a + 1,b);
    }
```

(a) No

(b) Yes

16. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a > b) return 0;
    else return a * prod(a + 1,b);
    }
```

(a) No

(b) Yes

17. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a == b) return a;
    else return b * prod(a,b-1);
    }
```

(a) No

(b) Yes

18. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a == b) return a;
    else return a * prod(a,b-1);
    }
```

(a) Yes

(b) No

19. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
def prod(int a,int b)
    {
    if (a > b) return 0;
    else return b * prod(a,b-1);
    }
```

(a) Yes

(b) No

20. Consider the problem statement: *product the numbers from* a *to* b (*inclusive*). Assuming $a$ is less than or equal to $b$, does this recursive function compute the correct result?

```
int prod(int a,int b)
    {
    if (a > b) return b;
    else return b * prod(a,b-1);
    }
```

(a) Yes

(b) No

**Concept:** *understanding recurrences*

21. Consider this recurrence:

```
f(a,b) is 1 if b is zero
f(a,b) is a * f(a,b - 1) otherwise
```

The recurrence is implemented with:

(a) one function with an if and else if and else if and else

(b) one function with an if and else

(c) two functions, one with an if and else

(d) two functions, one with an if and else if and else

(e) one function with an if and else if and else

22. Consider this recurrence:

```
f(t,a,b) is t if b is zero
f(t,a,b) is f(a * t,a,b - 1) otherwise
```

The recurrence is implemented with:

(a) one function with an if and else

(b) one function with an if and else if and else if and else

(c) one function with an if and else if and else

(d) two functions, one with an if and else

(e) two functions, one with an if and else if and else

23. Consider this recurrence:

```
f(n) is 0 if n is 0
f(n) is 1 if n is 1
f(n) is f(n - 1) + f(n - 2) otherwise
```

The recurrence is implemented with:

(a) one function with an if and else if and else if and else

(b) one function with an if and else if and else

(c) one function with an if and else

(d) two functions, one with an if and else if and else

(e) two functions, one with an if and else

24. Consider this recurrence:

```
g(a,b,n) is a if n is 0
g(a,b,n) is g(b,a + b,n - 1) otherwise
```

The recurrence is implemented with:

(a) one function with an if and else if and else

(b) one function with an if and else

(c) one function with an if and else if and else if and else

(d) two functions, one with an if and else

(e) two functions, one with an if and else if and else

25. Consider this recurrence:

```
f(t,a,b) is t if b is zero
f(t,a,b) is f(t,a * a,b / 2) if b is even
f(t,a,b) is f(t * a,a,b - 1) otherwise
```

The recurrence is implemented with:

(a) two functions, one with an if and else

(b) one function with an if and else if and else

(c) two functions, one with an if and else if and else

(d) one function with an if and else

(e) one function with an if and else if and else if and else

26. Consider this recurrence:

```
f(a,b) is 1 if b is zero
f(a,b) is f(a * a,b / 2) if b is even
f(a,b) is a * f(a,b - 1) otherwise
```

The recurrence is implemented with:

(a) one function with an if and else if and else

(b) two functions, one with an if and else

(c) one function with an if and else

(d) two functions, one with an if and else if and else

(e) one function with an if and else if and else if and else

27. Consider this recurrence:

```
g(t,a,b) is t if b is zero
g(t,a,b) is g(t,a * a,b / 2) if b is even
g(t,a,b) is g(t * a,a,b - 1) otherwise
```

The recurrence is implemented with:

(a) one function with an if and else if and else

(b) two functions, one with an if and else

(c) one function with an if and else if and else if and else

(d) one function with an if and else

(e) two functions, one with an if and else if and else

**Concept:** *implementing recurrences*

28. Implement this recurrence:

```
f(a,b) is 1 if b is zero
f(a,b) is a * f(a,b - 1) otherwise
```

29. Implement this recurrence:

```
f(t,a,b) is t if b is zero
f(t,a,b) is f(a * t,a,b - 1) otherwise
```

30. Implement this recurrence:

```
f(n) is 0 if n is 0
f(n) is 1 if n is 1
f(n) is f(n - 1) + f(n - 2) otherwise
```

31. Implement this recurrence:

```
g(a,b,n) is a if n is 0
g(a,b,n) is g(b,a + b,n - 1) otherwise
```

32. Implement this recurrence:

```
f(t,a,b) is t if b is zero
f(t,a,b) is f(t,a * a,b / 2) if b is even
f(t,a,b) is f(t * a,a,b - 1) otherwise
```

33. Implement this recurrence:

```
f(a,b) is 1 if b is zero
f(a,b) is f(a * a,b / 2) if b is even
f(a,b) is a * f(a,b - 1) otherwise
```

34. Implement this recurrence:

```
g(t,a,b) is t if b is zero
g(t,a,b) is g(t,a * a,b / 2) if b is even
g(t,a,b) is g(t * a,a,b - 1) otherwise
```

35. Implement this recurrence: (another power function)

```
power2(b,e) is 1 if e is zero
power2(b,e) is power2(b*b,e/2) if b is even
power(b,e) is b * power2(b,e - 1) otherwise
```

36. Implement this recurrence: (countEvens function)

```
countEvens(array,size) is 0 if array is empty
countEvens(array,size) is 1 + countEvens(tail(array)) if head(array) is even.
countEvens(array,size) is countEvens(tail(array)) if otherwise
```

37. Implement this recurrence: (prime function)

```
prime(n) is nodiv(n,2)
nodiv(n,current) is 1 if current > sqrt(n)
nodiv(n,current) is 0 if current divides n with zero remainder
nodiv(n,current) is nodiv(n,current + 1) otherwise
```

38. Implement this recurrence: (fibonacci function)

```
fib(n) is 0, if n is 0,
fib(n) is 1, if n is 1,
fib(n) is fib(n-1) + fib(n-2), otherwise
```

39. Implement the following recurrences (mult function)

```
mult(a,b) is 0, if b = 0
mult(a,b) is a, if b = 1
mult(a,b) is a + mult(a,b-1), otherwise
```

**Concept:** *implementing recursion*

40. Implement a recursive function (print(int array[], int size)), which will print all elements of the array.

41. Implement a recursive function (print(int array[], int size)), which will print all elements of the array in reverse.

42. Implement a recursive function (print(char array[], int size)), which will print all character of the array in reverse.

43. Implement a recursive function (print(char array[], int size)), which will print all character of the array.

44. Implement a recursive function (sum(int array[], int size)), which returns the sum of all elements in an array.

45. Implement a recursive function (fibonacci(int n), which prints a series of n fibonacci numbers.

```
For example:
fibonacci(1) is 1
fibonacci(2) is 1
fibonacci(3) is 2
fibonacci(4) is 3
```

46. Implement a recursive function (isPalindrome(char *str, int size), which prints whether the string passed as 'str' variable is a palindrome or not.

```
For example:
isPalindrome("abba",4) is Palindrome
isPalindrome("abcba",5) is Palindrome
isPalindrome("abc",3) is Not A Palindrome
```

47. Implement a recursive function (reverse(char *str, int size), which prints the string in reverse.

```
For example:
reverse("abcde",5) results in "edcba"
reverse("happy",5) results in "yppah"
```

48. Implement a recursive function (series) which will sum the series as shown below. For example: series(n) = 1/1 + 1/2 + 1/4 + 1/8 + 1/n

49. Implement a recursive function (series) which will sum the series as shown below. For example: series(n) = $1/1 + 1/2 + 1/\ 2^2 + 1/\ 2^3 + .....+1/\ 2^n$