QUESTION 1

Marks Scored: 0 | Correct Answer: +5 Marks

The function f is defined as follows:

C/C++ Code

```
int f (int n) {
   if (n <= 1) return 1;
   else if (n % 2 == 0) return f(n/2);
   else return f(3n - 1);
}
```

Assuming that arbitrarily large integers can be passed as a parameter to the function, consider the following statements.

i. The function f terminates for finitely many different values of $n \geq 1$.
ii. The function f terminates for infinitely many different values of $n \geq 1$.
iii. The function f does not terminate for finitely many different values of $n \geq 1$.
iv. The function f does not terminate for infinitely many different values of $n \geq 1$.

Which one of the following options is true of the above?

- (i) and (iii)
- (i) and (iv)
- (ii) and (iii)
- (ii) and (iv)

EXPLANATION

The function terminates for all values having a factor of 2 {(2.x)2==0}
So, (i) is false and (ii) is TRUE.
Let n = 3, it will terminate in 2nd iteration.
Let n=5, it will go like 5 - 14 - 7 - 20 - 10 - 5 – and now it will repeat.
And any number with a factor of 5 and 2, there are infinite recursions possible.
So, (iv) is TRUE and (iii) is false.

Hence Option(D) is the correct Option.

QUESTION 2

Marks Scored: +5 Correct Answer: +5 Marks

Consider the code fragment written in JAVA below :

C/C++ Code

```
void f (int n)
{
 if (n <=1) {
  System.out.print(n);
 }
 else {
  f (n/2);
  System.out.print(n%2);
 }
}
```

What does f(173) print?

- ( ) 010110101
- ( ) 010101101
- ( ) 10110101
- (•) 10101101

EXPLANATION

$(173)_2$ = 10101101 . Here the function will print the binary representation of 173.

Hence Option(D) is the correct answer.

This contest is open for practice, you can submit and practice questions, if you attempted the contest you can view your submissions by clicking on "Reveal Answers Button"

QUESTION 3

Marks Scored: +3 Correct Answer: +3 Marks

Let $T(n)$ be a function defined by the recurrence $T(n) = 2T(n/2) + \sqrt{n}$ for $n \geq 2$ and $T(1) = 1$ Which of the following statements is TRUE?

- ○ $T(n) = \theta(\log n)$
- ○ $T(n) = \theta(\sqrt{n})$
- ◉ $T(n) = \theta(n)$
- ○ $T(n) = \theta(n \log n)$

EXPLANATION

$n^{(\log_2 2)}$ = n which is = n^(1-.5) = O(sqrt n) then by applying case 1 of master method we get T(n) = Θ(n) Please refer http://www.geeksforgeeks.org/analysis-algorithm-set-4-master-method-solving-recurrences/ for more details.

QUESTION 4

Marks Scored: +1 CorrectAnswer:+5 Marks

Consider the following C function:

C/C++ Code

```
int f(int n)
{
  static int i = 1;
  if (n >= 5)
    return n;
  n = n+i;
  i++;
  return f(n);
}
```

The value returned by f(1) is

- ○ 5
- ○ 6
- ◉ 7
- ○ 8

EXPLANATION

Let's understand the problem with given Dry Run:

Initially, the static variable i is initialized to 1.

f(1) is called:

- n is 1, which is less than 5.
- n is updated to n + i, which becomes 2.
- i is incremented to 2.
- The function calls itself recursively with the updated value of n, so it becomes f(2).

QUESTION 5

Marks Scored: +1 Correct Answer +5 Marks

Consider the following C function.

C/C++ Code

```
int fun (int n){
 int x = 1, k;
 if (n == 1)
   return x;
 for (k = 1; k < n; ++k)
   x = x + fun(k) * fun(n - k);
 return x;
}
```

The return value of fun(5) is __________.

- 0
- 26
- 51
- 71

EXPLANATION

```
fun(5) = 1 + fun(1) * fun(4) + fun(2) * fun(3) +
             fun(3) * fun(2) + fun(4) * fun(1)
       = 1 + 2*[fun(1)*fun(4) + fun(2)*fun(3)]

Substituting fun(1) = 1
       = 1 + 2*[fun(4) + fun(2)*fun(3)]

Calculating fun(2), fun(3) and fun(4)
fun(2) = 1 + fun(1)*fun(1) = 1 + 1*1 = 2
fun(3) = 1 + 2*fun(1)*fun(2) = 1 + 2*1*2 = 5
fun(4) = 1 + 2*fun(1)*fun(3) + fun(2)*fun(2)
       = 1 + 2*1*5 + 2*2 = 15

Substituting values of fun(2), fun(3) and fun(4)
fun(5) = 1 + 2*[15 + 2*5] = 51
```

Hence Option(C) is the correct answer.

QUESTION 6

Marks Scored: 0 | Correct Answer: +1 Marks

Predict the Output:

C/C++ Code

```
class GFG
{
 static int f(int a[],int i, int n)
 {
  if(n <= 0) return 0;
  else if(a[i] % 2 == 0) return a[i] + f(a, i+1, n-1);
  else return a[i] - f(a, i+1, n-1);
 }
  public static void main(String args[])
  {
   int a[] = {12, 7, 13, 4, 11, 6};
   System.out.print(f(a,0,6));
  }
}
```

- -9
- 5
- 15
- 19

EXPLANATION

f() is a recursive function which adds f(a+1, n-1) to *a if *a is even. If *a is odd then f() subtracts f(a+1, n-1) from *a. See below recursion tree for execution of f(a, 6). .

```
f(add(12), 6) /*Since 12 is first element. a contains address of 12 */
    |
    |
12 + f(add(7), 5) /* Since 7 is the next element, a+1 contains address of 7 */
       |
       |
   7 - f(add(13), 4)
            |
            |
        13 - f(add(4), 3)
```

This contest is open for practice, you can submit and practice questions, if you attempted the contest you can view your submissions by clicking on "Reveal Answers Button"

QUESTION 7

Marks Scored: +1 Correct Answer: +5 Marks

What will be the output of the following JAVA program?

C/C++ Code

```
class GFG
{
  static int d=1;
  static void count(int n)
  {
   System.out.print(n+" ");
   System.out.print(d+" ");
   d++;
   if(n > 1) count(n-1);
   System.out.print(d+" ");
 }

 public static void main(String args[])
   {
     count(3);
   }
}
```

- (•) 3 1 2 2 1 3 4 4 4
- ( ) 3 1 2 1 1 1 2 2 2
- ( ) 3 1 2 2 1 3 4
- ( ) 3 1 2 1 1 1 2

EXPLANATION

count(3) will print value of n and d. So 3 1 will be printed and d will become 2.

Then count(2) will be called. It will print value of n and d. So 2 2 will be printed and d will become 3.

Then count(1) will be called. It will print value of n and d. So 1 3 will be printed and d will become 4.

Now count(1) will print value of d which is 4. count(1) will finish its execution.

This contest is open for practice, you can submit and practice questions, if you attempted the contest you can view your submissions by clicking on "Reveal Answers button"

QUESTION 8

Marks Scored: 0 | Correct Answer: +1 Marks

Let T(n) be defined by T(1) = 10 and T(n + 1) = 2n + T(n) and for all integers $n \geq 1$. Which of the following represents the order of growth of T(n) as a function of

- O(n)
- O(n log n)
- $O(n^2)$
- $O(n^3)$

EXPLANATION

```
T(n + 1) = 2n + T(n)
By substitution method:
T(n + 1) = 2n + (2(n-1) + T(n-1))
T(n + 1) = 2n + (2(n-1) + (2(n-2) + T(n-2)))
T(n + 1) = 2n + (2(n-1) + (2(n-2) + (2(n-3) + T(n-3))))
T(n + 1) = 2n + 2(n-1) + 2(n-2) + 2(n-3)......2(n-(n-1) + T(1))
T(n + 1) = 2n + 2n - 2 + 2n - 4 + 2n - 6 +.... + 10
T(n + 1) = 2[n + n + n + ...] - 2[1 + 2 + 3 +...]
T(n + 1) = 2[n*n] - 2[n(n+1)/2]
T(n + 1) = 2[n*n] - [n*n + n]
T(n + 1) = n*n - n
T(n + 1) = O(n²)
```

QUESTION 9

Marks Scored: +1 Correct Answer:+5 Marks

Consider the below program, what operation is performed below:

C/C++ Code

```
void fun(int arr[], int n)
{

    if (n == 1)
        return;

    int count = 0;
    for (int i=0; i arr[i+1]){
            swap(arr[i], arr[i+1]);
            count++;
        }
        return;
    fun(arr, n-1);
}
```

- Insertion Sort Recursively
- Bubble Sort Recursively
- Selection Sort Recursively
- None

EXPLANATION

The above code is the Recursive implementation of the Bubble sort.

Hence Option (B) is the correct answer.

This contest is open for practice, you can submit and practice questions, if you attempted the contest you can view your submissions by clicking on "Reveal Answers Button"

QUESTION 10

Marks Scored: 0 | Correct Answer +5 Marks

Consider the JAVA function given below.

C/C++ Code

```
static int f(int j){
 int i = 50;
 int k;
 if (i == j){
  System.out.print("something");
  k = f(i);
  return 0;
 }
 else return 0;
}
```

Which one of the following is TRUE?

- The function returns 0 for all values of j.
- The function prints the string something for all values of j.
- The function returns 0 when j = 50.
- The function will exhaust the runtime stack or run into an infinite loop when j = 50

EXPLANATION

When j is 50, the function would call itself again and again as neither i nor j is changed inside the recursion.

Hence Option (D) is the correct answer.

This contest is open for practice, you can submit and practice questions, if you attempted the contest you can view your submissions by clicking on "Reveal Answers Button"

QUESTION 11

Marks Scored: +5 Correct Answer +5 Marks

What is the output of the following recursive function when called with n=5?

C/C++ Code

```
int foo(int n) {
    if(n == 0)
   return 1;
    else
   return n * foo(n - 1);
}
```

- 1
- 5
- 120
- 720

EXPLANATION

The given recursive function calculates the factorial of a non-negative integer n.

When the function is called with n=5, the function checks whether n is equal to 0. Since n is not equal to zero, it returns the value of n times the value of the same function called with n-1 as input. This recursive call continues until the base case (n==0) is reached.

The calculation for foo(5) would be: foo(5) = 5 * foo(4) foo(4) = 4 * foo(3) foo(3) = 3 * foo(2) foo(2) = 2 * foo(1) foo(1) = 1 * foo(0) foo(0) = 1

Substituting foo(0) = 1 in foo(1): foo(1) = 1 * 1 = 1 Substituting foo(1) = 1 in foo(2): foo(2) = 2 * 1 = 2 Substituting foo(2) = 2 in foo(3): foo(3) = 3 * 2 = 6 Substituting foo(3) = 6 in foo(4): foo(4) = 4 * 6 = 24 Substituting foo(4) = 24 in foo(5): foo(5) = 5 * 24 = 120

Therefore, the output of the function when called with n=5 is 120, which is option C.

QUESTION 12

Marks Scored: +5 Correct Answer +5 Marks

Let f(n) and g(n) be asymptotically non-negative functions. Which of the following is correct?

- ○ θ (f (n)*g(n)) = min (f (n), g(n))
- ○ θ (f (n)*g(n)) = max (f (n), g(n))
- ○ θ(f (n) + g(n)) = min (f (n), g(n))
- ◉ θ (f (n) + g(n)) = max (f (n), g(n))

EXPLANATION

- **Case-1:** When none of the f(n) and g(n) are constant functions - In this case max(f(n) , g(n)) <= f(n) * g(n) so max(f(n), g(n)) can not provide a upper bound for f(n) * g(n).
- **Case-2:** When both of the f(n) & g(n) are constant functions or when any one of the f(n) and g(n) is a non zero constant function, in this case f(n) * g(n) = theta(max(f(n), g(n))).
- **Case-3:** When at least any one of the f(n) and g(n) is 0. In this case f(n) * g(n) != theta(max(f(n), g(n))). Since max(f(n), g(n)) COULD BE unable to give a lower bound.

Option (D) is correct.

QUESTION 13

Marks Scored: +5 Correct Answer +5 Marks

Consider the same recursive C++ function that takes two arguments

C/C++ Code

```
unsigned int foo(unsigned int n, unsigned int r)
{
  if (n > 0)
    return (n % r + foo(n / r, r));
  else
    return 0;
}
```

What is the return value of the function foo when it is called as foo(513, 2)?

- 9
- 8
- 5
- 2

EXPLANATION

foo(513, 2) will return 1 + foo(256, 2). All subsequent recursive calls (including foo(256, 2)) will return 0 + foo(n/2, 2) except the last call foo(1, 2) . The last call foo(1, 2) returns 1. So, the value returned by foo(513, 2) is 1 + 0 + 0.... + 0 + 1. The function foo(n, 2) basically returns sum of bits (or count of set bits) in the number n.

Hence (D) is the correct Answer.

QUESTION 14

Marks Scored: +5 CorrectAnswer:+5 Marks

Consider the following recursive JAVA function. If get(6) function is being called in main() then how many times will the get() function be invoked before returning to the main()?

C/C++ Code

```
static void get (int n){
 if (n < 1)
   return;
 get(n - 1);
 get(n - 3);
 System.out.print(n);
}
```

- ○ 15
- ◉ 25
- ○ 35
- ○ 45

EXPLANATION

```
                              get(6) [25 Calls]
                              /
              [17 Calls] get(5)       get(3) [7 Calls]
                         /
                    get(4)    get(2)[5 Calls]
                    /
        [7 Calls] get(3)  get(1)[3 Calls]
                  /
             get(2)   get(0)
             /
[3 Calls]get(1)  get(-1)
      /
get(0) get(-2)
```

Hence Option (B) is the correct answer.

This contest is open for practice, you can submit and practice questions, if you attempted the contest you can view your submissions by clicking on "Reveal Answers Button"

QUESTION 15

Marks Scored: +1 Correct Answer: +5 Marks

Consider the following C-function:

C/C++ Code

```
double foo (int n){
   int i;
   double sum;
   if (n == 0) return 1.0;
   else{
      sum = 0.0;
      for (i = 0; i < n; i++)
         sum += foo (i);
      return sum;
   }
}
```

The space complexity of the above function is:

- O(1)
- O(n)
- O(n!)
- $O(n^n)$

EXPLANATION

Note that the function foo() is recursive. Space complexity is O(n) as there can be at most O(n) active functions (function call frames) at a time.