Quiz On Space And Time Complexity Answers With Explanation Wherever Possible

Q) Find the time complexity of the given code.

```
int count = 0;

for (int i = N; i > 0; i /= 2)

   for (int j = 0; j < i; j++)
      count++;
```

a) O(log N)
b) O(N)
c) O(N*log N)
d) O(N^2)

Answer is b) O(N)
Explanation: For a input integer n, the innermost statement of fun() is executed following times.

n + n/2 + n/4 + … 1

So time complexity T(n) can be written as

T(n) = O(n + n/2 + n/4 + … 1) = O(n)

The value of count is also n + n/2 + n/4 + .. + 1

Q) Find the Complexity of the given snippet.

```
void function(int n)
{
  int count = 0;
        //executes O(n) times
  for (int i=0; i<n; i++)
        //Executes 0(n^2) times
  for (int j=i; j< i*i; j++)
     if (j%i == 0)
     {
         //executes O(n^2) times
       for (int k=0; k<j; k++)
           System.out.println("*");
      }
  }
```

1) O(N^2)

2) O(N^5)

3) O(N^4)

4) O(N^3)

Answer is O(n^5)

Explanation: It is explained in the comments in the code.

Q3)

```
int a = 0, b = 0;
//executes n times
for (i = 0; i < N; i++) {
   a = a + Math.random();
}
//executes m times
for (j = 0; j < M; j++) {
   b = b + Math.random();
}
```

a)O(N * M) time, O(1) space
b)O(N + M) time, O(N + M) space
c)O(N + M) time, O(1) space
d)O(N + M) time, O(1) space

Answer is O(N+M) time and O(1) space

Explanation: The first loop is O(N) and the second loop is O(M). Since we don't know which is bigger, we say this is O(N + M). This can also be written as O(max(N, M)).
Since there is no additional space being utilized, the space complexity is constant / O(1)

Q4)

```
int a = 0, i = N;
while (i > 0) {
   a += i;
   i /= 2;
}
```

a) O(N)

b) O(Sqrt(N))

c) O(N / 2)

d) O(log N)

Answer is O(log N)

Explanation: We have to find the smallest x such that N / 2^N. Solving it by taking log to base will give x = log(N)

Q5) Find the Complexity of the given code.

```
 int gcd(int n, int m)
 {
    if (n%m ==0) return m;
    if (n < m) swap(n, m);
       while (m > 0)
```

```
    {
      n = n%m;
      swap(n, m);
    }
  return n;
}
```

1) Θ(logn)
2) Ω(n)
3) Θ(loglogn)
4) Θ(sqrt(n))

Answer is 1) O(logN)

Explanation: Above code is implementation of the Euclidean algorithm for finding Greatest Common Divisor (GCD).

Q6) Frequently, the memory space required by an algorithm is a multiple of the size of input. State if the statement is True or False or Maybe.

1) TRUE
2) FALSE
3) May be
4) None of the Above

Answer is 1) True

Q7 For many problems such as sorting, there are many choices of algorithms to use, some of which are extremely___.

a) Space efficient
b) Time efficient
c) Both (a) and (b)
d) None of the above

Answer c) It is both space and time efficient

Q8) To verify whether a function grows faster or slower than the other function, we have some asymptotic or mathematical notations, which is_.

a) Big Omega Ω (f)
b) Big Theta θ (f)
c) Big Oh O (f)
d) All of the above

Answer is d) All of the above as all of the above are defined mathematical notation

Q9) An algorithm that indicates the amount of temporary storage required for running the algorithm, i.e., the amount of memory needed by the algorithm to run to completion is termed as_.

a) Big Theta θ (f)
b) Space complexity
c) Big Oh O (f)
d) Big Oh O (f)
e) None of the above

Answer b) Space Complexity

Q10) ___algorithm is one which utilizes minimum processor time and requires minimum memory space during its execution.

a) Best
b) Efficient
c) Both (a) and (b)
d) None of the above

Answer is c) both Best and Efficient

Q11) lg (n!) = ……..

a) O(n)
b) O(lg n)
c) O(n^2)
d) O(n lg n)

Answer is d) O(n lgn)

Q12) The running time of an algorithm is given by T(n) = T(n - 1) + T(n - 2) - T(n - 3), if n > 3 n, otherwise.

a) N
b) log n
c) n+1
d) None of these

Answer is a O(N)

Explanation: As

T(n) = T(n - 1) + T(n - 2) - T(n - 3)

T(n - 1) = T(n - 2) + T(n - 3) - T(n - 4)

T(n - 2) = T (n - 3) + T(n - 4) - T(n - 5)

T(n - 3) = T (n - 4) + T(n - 5) - T(n - 6)

T(n - 4) = T (n - 5) + T(n - 6) - T(n - 7)

……

……

T(6) = T(5) + T(4) - T(3)

T(5) = T(4) + T(3) - T(2)

T(4) = T(3) + T(2) - T(1)

On Adding all the above recurrences, all the red terms of a recurrence will be cancelled out with the red terms of consecutive recurrences, similarly all the blue terms of a recurrence will be cancelled out with the blue terms of consecutive recurrences.

Thus we get

T(n) = T(n - 2) + T(3) - T(1),

but since T(3) =3 & T(1) = 1

so our final recurrence will be T(n) = T(n - 2) + 2

Solution of T(n) = T(n - 2) + 2 will be n.

Q13)The concept of order (Big O) is important because

1) it can be used to decide the best algorithm that solves a given problem
2) it determines the maximum size of a problem that can be solved in a given system, in a given amount of time
3) all of the above
4) none of these

Answer is 3) all of the above

Q14) Find complexity of the following

```
void pat(){
//executes log (input size) times
  for (int i = 2; i <=n; i = pow(i, c)) {
    System.out.print("*")
  }

//Here fun is sqrt or cuberoot or any other constant root
//executes log n times
  for (int i = n; i > 1; i = fun(i)) {
   System.out.print("*")
  }
}
```

a) O(N LogN)
b) O(N^3)
c) O(N^2)
d) LogLog N

Answer is d) LogLogN

Explanation: Explained in comments in code.

Q15)

```
int a = 0;
//executes O(n) times
   for (i = 0; i < N; i++) {
//executes O(n) times

       for (j = N; j > i; j--) {
          a = a + i + j;
       }
   }
```

A) O(N)
B) O(N*log(N))
C) O(N * Sqrt(N))
D) O(N*N)

Explanation: explained as comments in code.