**1. What is the time, space complexity of**the **following code:**

|  |
| --- |
| **int** a = 0, b = 0;  **for** (i = 0; i < N; i++) {      a = a + **rand**();  }  **for** (j = 0; j < M; j++) {      b = b + **rand**();  } |

**Options:** 

1. O(N \* M) time, O(1) space
2. O(N + M) time, O(N + M) space
3. O(N + M) time, O(1) space
4. O(N \* M) time, O(N + M) space

Ans: 3. O(N + M) time, O(1) space

**2. What is the time complexity of**the **following code:**

|  |
| --- |
| **int** a = 0;  **for** (i = 0; i < N; i++) {  **for** (j = N; j > i; j--) {          a = a + i + j;      }  } |

**Options:** 

1. O(N)
2. O(N\*log(N))
3. O(N \* Sqrt(N))
4. O(N\*N)

**Ans:** 4. O(N\*N)

**3. What is the time complexity of**the **following code:**

|  |
| --- |
| **int** i, j, k = 0;  **for** (i = n / 2; i <= n; i++) {  **for** (j = 2; j <= n; j = j \* 2) {          k = k + n / 2;      }  } |

**Options:**

1. O(n)
2. O(nLogn)
3. O(n^2)
4. O(n^2Logn)

**Ans:** 2. O(nLogn)

Because inner loop will run logn time because j is being double in every iteration and outer loop will run n/2 times.

**4. What does it mean when we say that an algorithm X is asymptotically more efficient than Y?**   
**Options:** 

1. X will always be a better choice for small inputs
2. X will always be a better choice for large inputs
3. Y will always be a better choice for small inputs
4. X will always be a better choice for all inputs

2. X will always be a better choice for large inputs

**5. What is the time complexity of**the **following code:**

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

**Options:** 

1. O(N)
2. O(Sqrt(N))
3. O(N / 2)
4. O(log N)

Ans: 4. O(log N)

**6. Which of the following best describes the useful criterion for comparing the efficiency of algorithms?**

1. Time
2. Memory
3. Both of the above
4. None of the above

Ans: 3. Both of the above

**7. How is time complexity measured?**

1. By counting the number of algorithms in an algorithm.
2. By counting the number of primitive operations performed by the algorithm on given input size.
3. By counting the size of data input to the algorithm.
4. None of the above

Ans: 2. By counting the number of primitive operations performed by the algorithm on given input size.

|  |
| --- |
|  |