1. What is the time complexity of the following code snippet?

for (int i = 0; i < n; i++) {

for (int j = 1; j < n; j \*= 2) {

cout << i << " " << j << endl;

}

}

a) O(n)

b) O(n^2)

c) O(log n)

**d) O(n log n)**

2. Space complexity measures:

a) The number of variables used in a program.

**b) The amount of memory space used by a program to solve a problem.**

c) The number of operations performed in a program.

d) The efficiency of a program in terms of execution time.

3. What is the space complexity of the following code snippet?

int sum = 0;

for (int i = 0; i < n; i++) {

sum += i;

}

**a) O(1)**

b) O(n)

c) O(n^2)

d) O(log n)

4. The worst-case time complexity of an algorithm:

a) Represents the average performance of the algorithm.

**b) Represents the maximum performance time required by the algorithm.**

c) Represents the minimum performance time required by the algorithm.

d) Is the same as the average-case time complexity.

5. What is the output of the following code snippet?

int n = 5;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= i; j++) {

cout << j << " ";

}

}

a) 1 2 3 4 5

b) 1 2 3 4 5 1 2 3 4 5

**c) 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5**

d) 1 2 3 4 5 2 3 4 5 3 4 5 4 5 5

6. Omega (Ω) notation is used to represent:

a) The average-case time complexity of an algorithm.

**b) The lower bound of an algorithm's time complexity.**

c) The exact time complexity of an algorithm.

d) The best-case time complexity of an algorithm.

7. What is the time complexity of the following code snippet?

int result = 0;

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

result += i + j;

}

}

a) O(n)

**b) O(n^2)**

c) O(n^3)

d) O(log n)

8. Which of the following statements is true regarding the relationship between Big O, Omega, and Theta notation?

**a)Big O is an upper bound, Omega is a lower bound, and Theta is a tight bound.**

b) Big O is a lower bound, Omega is an upper bound, and Theta is a tight bound.

c) Big O, Omega, and Theta notations all represent the same complexity.

d) Big O represents the lower bound, and Omega represents the upper bound.

9. What is the space complexity of the following code snippet?

int\* arr = new int[n];

for (int i = 0; i < n; i++) {

arr[i] = i;

}

delete[] arr;

a) O(1)

**b) O(n)**

c) O(n^2)

d) O(log n)

10. The best-case time complexity of an algorithm is also known as:

a) Lower bound

**b) Best bound**

c) Upper bound

d) Tight bound

11. What is the output of the following code snippet?

int x = 5;

cout << x++ << " " << ++x << " " << x << endl;

**a) 5 7 7**

b) 6 7 7

c) 6 7 6

d) 5 6 6

12. In the context of time complexity analysis, "n" represents:

**a) The size of the input data.**

b) The number of operations performed.

c) The efficiency of the algorithm.

d) The number of variables used.

13. What is the time complexity of the following code snippet?

int i = n;

while (i > 0) {

cout << i << endl;

i /= 2;

}

a) O(n)

**b) O(log n)**

c) O(n log n)

d) O(n^2)

14. If an algorithm's time complexity is Θ(n log n), it means that:

a) The algorithm's running time is always exactly n log n.

**b) The algorithm's running time grows with the product of n and log n.**

c) The algorithm has constant time complexity.

d) The algorithm's running time decreases with larger input sizes.

15. What is the space complexity of the following code snippet?

void recursiveFunc(int n) {

if (n <= 0) {

return;

}

cout << n << endl;

recursiveFunc(n - 1);

}

a) O(1)

b) O(n)

**c) O(log n)**

d) O(n^2)

16. If an algorithm's time complexity is O(1), it means that:

a) The algorithm has a constant running time.

b) The algorithm has a logarithmic running time.

**c) The algorithm's running time is independent of the input size.**

d) The algorithm's running time grows linearly with the input size.

17. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr;

cout << \*(ptr + 2) << endl;

a) 1

b) 2

**c) 3**

d) 4

18. The notation used to represent the average-case time complexity of an algorithm with input size "n" is:

a) O(n)

**b)θ(n)**

c) O(1)

d) Ω(n)

19. What is the time complexity of the following code snippet?

for (int i = 0; i < n; i++) {

cout << "Hello ";

}

for (int j = 0; j < n; j++) {

cout << "World ";

}

**a) O(n)**

b) O(n^2)

c) O(2n)

d) O(1)

20. The notation used to represent the space complexity of an algorithm is:

a) O(n)

b) Ω(n)

**c) O(f(n))**

d) θ