**Question 1:**

**What is the time complexity of the following code snippet?**

for i in range(n): print("Hello, World!")

a) O(1)  
b) O(n)  
c) O(log n)  
d) O(n^2)

**Question 2:**

**What is the space complexity of the following code snippet?**

def find\_max(arr): max\_value = arr[0] for element in arr: if element > max\_value: max\_value = element return max\_value

a) O(1)  
b) O(n)  
c) O(log n)  
d) O(n^2)

**Question 3:**

**What is the time complexity of the following code snippet?**

for i in range(n): for j in range(n): print(i, j)

a) O(1)  
b) O(n)  
c) O(n^2)  
d) O(log n)

**Question 4:**

**What is the space complexity of the following code snippet?**

def fibonacci(n): fib = [0] \* (n+1) fib[0], fib[1] = 0, 1 for i in range(2, n+1): fib[i] = fib[i-1] + fib[i-2] return fib[n]

a) O(1)  
b) O(n)  
c) O(log n)  
d) O(n^2)

**Question 5:**

**What is the time complexity of the binary search algorithm?**

a) O(1)  
b) O(log n)  
c) O(n)  
d) O(n^2)

**Solution:** b) O(log n)

**What does Big O notation represent in terms of algorithm analysis?**

a) Best-case time complexity  
b) Average-case time complexity  
c) Worst-case time complexity  
d) Exact running time

**Question 6:**

**If a function f(n) is O(g(n)), what does it imply?**

a) f(n) grows faster than g(n)  
b) f(n) grows at the same rate as g(n)  
c) f(n) grows slower than g(n)  
d) No relationship between f(n) and g(n)

**Question 7:**

**What does Big Omega (Ω) notation represent in terms of algorithm analysis?**

a) Best-case time complexity  
b) Average-case time complexity  
c) Worst-case time complexity  
d) Lower bound on the growth rate of a function

**Question 8:**

**If f(n) = Θ(g(n)), what does it mean?**

a) f(n) grows faster than g(n)  
b) f(n) grows at the same rate as g(n)  
c) f(n) grows slower than g(n)  
d) No relationship between f(n) and g(n)

**Question 9:**

**Which of the following statements is true regarding Big O notation?**

a) f(n) = O(g(n)) implies g(n) = O(f(n))  
b) f(n) = O(g(n)) implies g(n) = Ω(f(n))  
c) f(n) = Θ(g(n)) implies g(n) = O(f(n))  
d) f(n) = Ω(g(n)) implies g(n) = Θ(f(n))

**Question 1: Solution:** b) O(n)

**Question 2: Solution:** O(1)

**Question31: Solution:** c) O(n^2)

**Question 4:** O(n)

**Question 5: Solution:** b) O(log n)

**Question 6: Solution:** b) O(n)

**Question 7: Solutio** Worst-case time complexity

**Question 8:** f(n) grows at the same rate as g(n)

**Question 9:** f(n) = Θ(g(n)) implies g(n) = O(f(n))