Q1. Consider the definition:
   \[ T(n) = O(f(n)) \] if there exists an integer \( n_0 \) and a constant \( c > 0 \) such that for all integers \( n > n_0 \), we have \( T(n) \leq c \cdot f(n) \).

Find values for \( n_0 \) and \( c \) to prove the following are big-oh relationships.

1. Prove \( 8 \cdot n^3 - 9 \cdot n \) is \( O(n^3) \)
2. Prove \( \log_2 n + 20 \) is \( O(\log_2 n) \)
3. Prove \( \log_2 n + n \) is \( O(n) \)

Q2. Give a big-oh upper bound on the running time of the trivial while-loop
   ```
   while (C) {} where C is a condition that does not involve any function calls.
   ```

Q3. Give a big-oh upper bound on the running time of the for-loop that includes function \( f(n) \) whose big-oh upper bound is \( O(f(n)) \).
   ```
   for (int i = 1; i <= n; i++)
   f(n);
   ```

Q4. Find a big-oh upper bound for the worst-case time required by the following algorithm:
   ```
   bool iskey(int s[], int n, int key) {
   for (int i = 0; i < n - 1; i++)
   for (int j = i + 1; j < n; j++)
   if (s[i] + s[j] == key)
   return true;
   else
   return false;
   }
   ```

Q5. What is the order of each of the following tasks in the worst case?

   1. Computing the sum of the first \( n \) even integers by using a for loop
   2. Displaying all \( n \) integers in an array
   3. Displaying one array element
   4. Inserting one element from an array list

Q6. Binary search needs a list to be sorted before searching for a particular item. Given an \( n \)-element unsorted list, you sort the list using an \( n \log n \) efficient sorting algorithm and thereafter apply binary search for this list. Give the order of this sequence of work and compare it with sequential search that can be used directly for an unsorted list.
Q7. For large arrays and in the worst case, is selection sort faster than insertion sort? Explain

Q8. The following fragment of code computes the matrix multiplication of a[n][n] and b[n][n]. Give a big-oh upper bound on the running time (as a function of n).

```c
for ( int i = 0, i < n, i++ )
    for ( int j = 0, j < n, j++ ) {
        c[i][j] = 0.0;
        for ( int k = 0, k < n, k++ )
            c[i][j] += a[i][k] * b[k][j];
    }
```

Q9. Give an analysis of the running time for the following segments of code.

```c
i = 1;
while ( i <= 2n ) {
    x++;
    i += 2;
}
for ( i = 1; i <= n; i++ )
    for ( j = 1; j <= i; j++ )
        for ( k = 1; k <= j; k++ )
            x++;

i = n;
while ( i >= 1 ) {
    for ( j = 1; j <= n; j++ )
        x++;
    i /= 2;
}
```

Q10. Find a formula for the sum of the first n odd positive integers. Using mathematical induction, prove it is the correct answer.

Q11. Show that \( \sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \ldots + n^2 = n(n + 1)(2n + 1)/6 \)

Q12. Find a formula for 1*2 + 2 * 3 + 3 * 4 + \ldots + n * (n + 1). Using mathematical induction, prove it is the correct answer. Hint: use the formula in Q2 and another favorite formula: \( \sum_{i=1}^{n} i = 1 + 2 + \ldots + n = n(n + 1)/2 \)

Q13. Prove that 7^n – 1 is divisible by 6, for = 1, 2, …

Q14. Show that postage of six cents or more can be achieved by using only 2-cent and 7-cent stamps.

Q15. Show that any (2i) * (3j) board, where i and j are positive integers, with no square missing, can be tiled with trominoes.

Q16. Suppose that n > 1 people are positioned so that each has a unique nearest neighbor. Suppose further that each person has a pie that is hurled at the nearest neighbor. A survivor
is a person that is not hit by a pie. Use induction on \( n \) to show that if \( n \) is odd, there is always at least one survivor.

Q17. Assume that a chocolate bar consists of \( n \) squares arranged in a rectangular pattern. The bar or a smaller rectangular piece of the bar can be broken along a vertical or a horizontal line separating the squares. Assuming that only one piece can be broken at a time, determine how many breaks you must successively make to break the bar into \( n \) separate squares. Use strong induction to prove your answer. (Text 4.2.10)

A few items to help with your exercise on big-oh

- \( 1 + 2 + 3 + \ldots + n = \frac{n(n + 1)}{2} \)
- \( 1 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(n+2)}{6} \)
- \( a + a \cdot r^1 + a \cdot r^2 + \ldots + a \cdot r^n = a(r^{n+1} - 1)/(r-1) \)