1. What is the running time for the following code, which multiplies two $n \times n$ matrices $A$ and $B$, storing the result in $C$? [4 points]

```java
for i=1 to n
    for j=1 to n {
        C[i,j] = 0
        for k=1 to n
            C[i,j] = C[i,j] + A[i,k]*B[k,j]
    }
```

2. Determine the run times of the following two pieces of code, which do pretty much nothing. [6 points]

```java
sum =0
for i = 1 to n*n*n
    for j=1 to i*i
        sum += (i-j)

and

sum =0
for i = 1 to n*n
    j=1
    while j<i
        sum++
        j *= 5
```

3. Show that $\sum_{i=1}^{n}[\log i] = \Theta(n \log n)$. Do this directly, without recourse to Stirling’s approximation [6 points]

4. Suppose that each row of an $n \times n$ array $A$ consists of 1’s and 0’s in such a way that, for any row, all the 1’s come before any 0. Assuming that $A$ is already in memory, describe how to find which row of $A$ which contains the most 1’s. Aim to do this in $O(n)$ time, not $O(n^2)$. [6 points]
5. Occasionally, multiplying the sizes of nested loops can give an over-estimate for the big-O running time. This happens when an innermost loop is infrequently executed. With this in mind, determine the running time of the following piece of code. [8 points]

```cpp
for( int i = 0; i < n; i++ )
    for( int j = 0; j < i * i; j++ )
        if( j % i == 0 )
            for( int k = 0; k < j; k++ )
                sum++;
```

6. (Recurrence Relations) Exercise 4.3-1, p 75. [6 points]

7. (Recurrence Relations) Exercise 4.3-2, p 75. [3 points]

8. Describe a non-recursive method for finding, by link hopping, the (approximate) middle node of a singly-linked list. This method must use only link hopping; it cannot use a counter. What is the running time of your method? [4 points]

9. (Reverse a linked list in constant space) Exercise 10.2-7, p 209 [8 points]

10. Describe how to implement the stack ADT using two queues. What is the running time of the push() and pop() methods in this case? [6 points]

Total: 57 points