CS 340 - Assignment 1

Exercise 1.5

```
int num(int N) {
        if (N == 1)
                return 1;
        else if (N % 2 == 0)
                return num(N/2);
        else
                return num(N/2) + 1;
}
```

Exercise 1.10

$2^{10} = 1024 \equiv 4\,(mod\,5)$
$2^{20} = 2^{10} * 2^{10} \equiv 4 * 2^{10} \equiv 4 * 4 = 16 \equiv 1\,(mod\,5)$
$2^{40} = 2^{20} * 2^{20} \equiv 1 * 2^{20} \equiv 1 * 1 \equiv 1\,(mod\,5)$
$2^{80} = 2^{40} * 2^{40} \equiv 1 * 2^{40} \equiv 1 * 1 \equiv 1\,(mod\,5)$
$2^{100} = 2^{80} * 2^{20} \equiv 1 * 2^{20} \equiv 1 * 1 \equiv 1\,(mod\,5)$

Therefore $2^{100}\,mod\,5 = 1$

Exercise 2.1

$37, \frac{2}{N}, \sqrt{N}, N, NloglogN, NlogN, Nlog(N^2), Nlog^2N, N^{1.5}, N^2, N^2logN, N^3, 2^{\frac{N}{2}}, 2^N$

Exercise 2.3

Let $f(x) = NlogN\ and\ g(x) = N^{1+\frac{\epsilon}{\sqrt{logN}}} = N * N^{\frac{\epsilon}{\sqrt{logN}}}$
Since N > 0 (otherwise logN is invalid), divide both f(x) and g(x) to N
$f(x) = logN$ (1)
$g(x) = N^{\frac{\epsilon}{\sqrt{logN}}}$ (2)
Get logarithm (base 2) on both functions:
$(1) = loglogN$
$(2) = \frac{\epsilon}{\sqrt{logN}}logN$
Multiply (1) and (2) to $\sqrt{logN}$
$(1) \Leftrightarrow \sqrt{logN} * loglogN$ (3)
$(2) \Leftrightarrow \epsilon * logN$ (4)

Get logarithm (base 2) on (3) and (4)

(3) $\Leftrightarrow \frac{1}{2}loglogN + logloglogN$ (5)

(4) $\Leftrightarrow log\epsilon + loglogN$ (6)

Subtract (5) and (6) to $\frac{1}{2}loglogN$

(5) $\Leftrightarrow logloglogN$ = F(x) $= O(logloglogN)$

(6) $\Leftrightarrow log\epsilon + \frac{1}{2}loglogN$ = G(x) = $O(loglogN)$

Since G(x) grows faster than F(x), g(x) grows faster than f(x)

Exercise 2.7

(1) There is one loop where i goes from 0 to n - 1. Therefore (1) = O(n)
Compare: The actual running time supports the analysis.

(2) There is a nested loop and both counter go to n. Therefore (2) = O(n^2)
Compare: The actual running time supports the analysis.

(3) There is a nested loop. In the inner loop, the counter goes from 0 to n^2. Therefore (3) = O(n^3)
Compare: The actual running time supports the analysis.

(4) There is a nested loop. In the inner loop, the counter j goes from 0 to i while i goes from 0 to n. Therefore (4) = O(n^2)
Compare: The actual running time supports the analysis.

(5) There are three loops inside each other. The counter j in the second loop goes from 0 to i^2 while i goes from 0 to n. The counter k in the third loop goes from 0 to j. Therefore (5) = O(n^4)
Compare: The actual running time supports the analysis.

(6) The counter i goes from 1 to n, j goes from 1 to i^2. However the second loop only runs when j % i == 0. Lastly, k is from 0 to j. Since the loop that has i only run n times, (6) = O(n^3)
Compare: The actual running time supports the analysis.

Exercise 2.10

a) In order to add two N-digit integers, we add each corresponding digit. Therefore running time is O(N).
b) To multiply 2 N-digit integers, we multiply each digit of one integer to the other, which takes N^2 times. After that, add the results together which takes N-2 time. Therefore running time for multiplication is N^2 + N - 2 or O(N^2)

c) In the worst case, the result is greater than 0. The running time for division is 1. In order to calculate the remainder, multiply the result to the second number which take N time and then subtract the first number to that result which takes another N time. Therefore running time for the operation is 1 + N + N = 2N + 1 or O(N)

Exercise 2.20

a) Code segment is submitted separately.
b) The worse case running time is when N is prime (For example 131) because the counter will have to run from 2 to $\sqrt{131}$ (converted to int) = 11 since the loop only stops when 131 % the counter = 0 (which doesn't happen since 131 is prime)
c) To convert a decimal to binary, we divide that decimal by 2 repeatedly until there is 1 or 0 left. Therefore if B is the number of bits in the binary representation of N, B = $\log_2 N + 1$
d) $B = log_2 N + 1 \Leftrightarrow B - 1 = log_2 N \Leftrightarrow N = 2^{B-1}$
Worse case running time in terms of N = $\sqrt{N}$
Worse case running time in terms of B = $\sqrt{2^{B-1}} = 2^{\frac{B-1}{2}}$
e) 20-bit prime: 993319 ———-> running time: $6 * 10^{-6}$ seconds
40-bit prime: 824633720831 ————> running time: 0.000179
(The screenshot will be provided separately. Running time is only measured when the input is prime for comparison. Otherwise, the loop will halt before i goes to $\sqrt{N}$)
f) It is more reasonable to give the running time in terms of N because the running time in terms of B is more complicated (2 to the power of B compare to square root of N).

Exercise 2.27

Initialize the starting point at the top right corner. In order to find X, first compare X to the current position in the matrix (starting point - top right corner), if X is greater than the value in the current position, move the current position down. Otherwise move the current position to the left. Since every values in the left of the current position is always less than the value in the current position and values below the current position are always greater, the current position can only be moved left or down, not both direction. Therefore, the running time of this algorithm is O(N).