Homework 2.


1. (a) (1pt)

```
int func1(int n) {
    int s = 0;
    int i = n;
    while ( i >= 0 ) {
        int j = n*n;
        while ( j >= 1) {
            j = j / 2;
            s = s + i*j;
        }
        i = i - 4;
    }
    return s;
}
```

$i = n, n-4, n-8, \cdots, 0 \quad \frac{n}{4}$

$j = n^2, \frac{1}{2}n^2, \frac{1}{4}n^2, \cdots, 1 \quad \log n^2 = 2\log n$

$T(n) = \frac{n}{4} * 2\log n = \frac{1}{2} n\log n = \text{Theta}(n\log n)$.

(b) (1 pt) Note: function func2 calls the function func1 defined in question 1(a).

```
int func2(int n) {
    int res = func1(n*n);
    if (res > 1000) {
        for (int k =0; k < n; k++) {
            res += func1(n);

        }
    }
    return res;
}
```

Since func1() takes $\Theta(n\log n)$ time.

Thus, res = func1(n*n) will take $\Theta(n^2 \log n)$ time.

The for loop variable k iterates n times.

thus the entire for loop takes $\Theta(n^2 \log n)$ time.

Summing them up, we get that $T(n) = \Theta(n^2 \log n)$.

2.
```
int recursive1(int n) {
      if (n > 1) {
        return 2 * recursive1(n-1);
      }
      else
        return n;
  }
```

(a). The function computes $\begin{cases} n, & n <= 1 \\ 2^{n-1}, & n > 1 \end{cases}$

when $n=1$, return 1

when $n=2$, return $2 * \text{recursive1}(1) = 2*1 = 2$

when $n=3$, return $2 * \text{recursive}(2) = 2 * 2 * \text{recursive}(1) = 2^2 * 1 = 2^2$

(b) base case: $T(1) = C_1$

recursive case: $T(n) = T(n-1) + C$

(c). $T(n) = T(n-1) + C$ (1)

$T(n-1) = T(n-2) + C$, substituting $T(n-1)$ in equation (1).

we have $T(n) = T(n-1) + C$

$= T(n-2) + C + C = T(n-2) + 2C$ (2)

Since $T(n-2) = T(n-3) + C$, substituting $T(n-2)$ in equation (2).

we have $T(n) = T(n-3) + C + 2C$

$= T(n-3) + 3C$

$\vdots$

we can write in this format: $T(n) = T(n-k) + kC$

Since $T(1) = C_1$.

Let $n-k=1$, we will get $k = n-1$.

Subsitituting into $T(n) = T(n-k) + kC$

gives that $T(n) = T(1) + (n-1)C$

$= nC - C + C_1$

$= \text{Theta}(n)$

---

3.

```
int recursive2(int n) {
   int s = 0;
   if (n <= 1)
        return 1;

   for (int i = 1; i <= n; i++) {
        s = s + Math.pow(2, i);
   }

   return recursive2(n/2) * recursive2(n/2) + s;
}
```

(1). $T(1) = C_1$

$T(n) = 2T(\frac{n}{2}) + n + C$ ( The for loop takes $\Theta(n)$ times).

(2). $T(n) = 2T(\frac{n}{2}) + n + C$

$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \frac{n}{2} + C$

Thus, $T(n) = 2(2T(\frac{n}{4}) + \frac{n}{2} + C) + n + C$

$= 4T(\frac{n}{4}) + n + 2C + n + C$

$= 4T(\frac{n}{4}) + 2n + 3C.$

$= 8T(\frac{n}{8}) + 3n + 7C.$

$T(n) = 2^k T(\frac{n}{2^k}) + kn + (2^k - 1)C$ stop at $T(1)$.

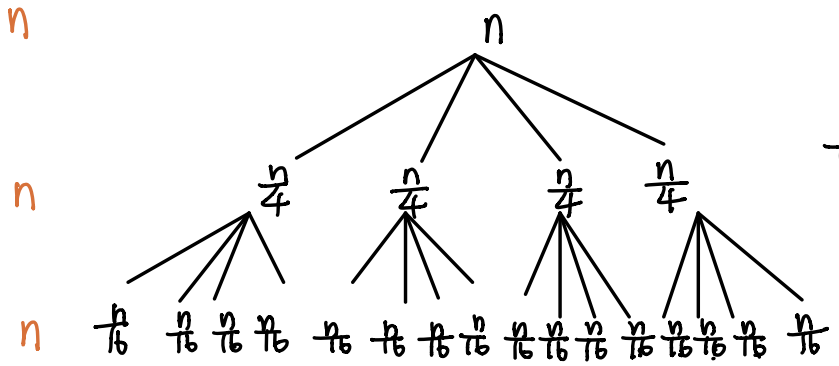
$\frac{n}{2^k} = 1. \Rightarrow n = 2^k. \quad k = \log n.$

$T(n) = nT(1) + n\log n + (n-1)C = (C+C_1)n + n\log n - C$

$T(n) = \text{Theta}(n \log n).$

4. $T(1) = c_1$

$T(n) = 4T(\frac{n}{4}) + n$




$T(\frac{n}{4}) = 4T(\frac{n}{16}) + \frac{n}{4}$.

$c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$ $c_1$

$4^{\log_4 n} = n$ leaves.

The height of the tree is $\log_4 n = \frac{\log_2 n}{\log_2 4} = \frac{1}{2}\log(n)$.

Summing it up: n leaves, the height of the tree is $\frac{1}{2}\log(n)$:

$c_1 * n + (\frac{1}{2}\log n - 1)n = \text{Theta}(n \log n)$.

5. 5. (1pt) Show the array 17, 10, 15, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3 after each pass of the **insertion** sort (after placing each element from the unsorted part of the list into the sorted part of the list). Underline the sorted part of the list. The first pass has already been done for you.
17, 10, 15, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3 Original array
10, 17, 15, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3 After the first pass

17, 10, 15, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3 Original array

10, 17, 15, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3 After the first pass

10, 15, 17, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3 After the second pas

10, 13, 15, 17, 4, 12, 7, 9, 16, 8, 5, 14, 3 After the third pass

4, 10, 13, 15, 17, 12, 7, 9, 16, 8, 5, 14, 3 After the 4th pass

4, 10, 12, 13, 15, 17, 7, 9, 16, 8, 5, 14, 3 After the 5th pass

4, 7, 10, 12, 13, 15, 17, 9, 16, 8, 5, 14, 3 After the 6th pass

4, 7, 9, 10, 12, 13, 15, 17, 16, 8, 5, 14, 3 After the 7th pass

4, 7, 9, 10, 12, 13, 15, 16, 17, 8, 5, 14, 3 After the 8th pass

4, 7, 8, 9, 10, 12, 13, 15, 16, 17, 5, 14, 3 After the 9th pass

4, 5, 7, 8, 9, 10, 12, 13, 15, 16, 17, 14, 3 After the 10th pass

4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 3 After the 11th pass

3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17 After the 12th pass

6. (2 pt) Show intermediate steps of sorting the array with Merge Sort. Show the result after each recursive call to the merge sort routine.
17, 10, 15, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3 Original array

10, 17, 15, 13, 4, 12, 7, 9, 16, 8, 5, 14, 3

10, 17, 13, 15, 4, 12, 7, 9, 16, 8, 5, 14, 3

10, 13, 15, 17, 4, 12, 7, 9, 16, 8, 5, 14, 3

10, 13, 15, 17, 4, 12, 7, 9, 16, 8, 5, 14, 3

10, 13, 15, 17, 4, 7, 12, 9, 16, 8, 5, 14, 3

4, 7, 10, 12, 13, 15, 17, 9, 16, 8, 5, 14, 3

4, 7, 10, 12, 13, 15, 17, 9, 16, 8, 5, 14, 3

4, 7, 10, 12, 13, 15, 17, 8, 9, 16, 5, 14, 3

4, 7, 10, 12, 13, 15, 17, 8, 9, 16, 5, 14, 3

4, 7, 10, 12, 13, 15, 17, 8, 9, 16, 3, 5, 14

4, 7, 10, 12, 13, 15, 17,3, 5, 8, 9, 14, 16

3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17