LAB 3A

Problem 1

$T(n) = T(n/2) + n; \quad T(1) = 1$

Soln :

$$T(n) \begin{cases} 1 & n = 1 \\ T(n/2) + n & \text{otherwise} \end{cases}$$

Here $a = 1$ $\quad c = 1$ $\quad k = 1$

$b = 2$ $\quad d = 1$

$$a < b^k = 1 < 2^1$$

$$\Theta(n^k) \Rightarrow \Theta(n)$$

Problem 2 :

```
Algorithm isPrime(n)
Input: a number n
Output: 1 if prime and 0 if not.
    $i \leftarrow 2$;
if $n < 2$ then return 1
If $n == 2$ then return 0
if $i * i > n$ then return
return isPrime(n);
```

Here, to find ~~out~~ if the given number n is prime or not we recursively call the method until $i * i > n$ which $\approx i * i - 1 = n$ $\quad$ Ex $= 5 * 5 = 25$ $\quad \sqrt{25} = 5$.

$\therefore \sqrt{n}$ -> steps to know if a given number is prime

$T(n) = O(\sqrt{n}) \Rightarrow O(n)$

$T(b) = O(f(2^b)) = O(\sqrt{2^b})$

problem 4: Sorting.

A - prove the algorithm is correct.

- valid Recursion → Base case is when the list L has 0 or 1 element. Self-calls reduce input size by half, so it lead to base case.
- Base case correct → If list has 0 or 1 element it is returned because it's already sorted.
- Recursive step → Assuming ~~[illegible]~~ recSort is correct for list of length < n, when we run recSort on list of length n, the algorithm produces partitioned sublists L_1 & L_2 of smaller length. So recSort ~~[illegible]~~ correctly sorts each.

B. use the version of master formula to show the running time.

$T(1) = 1$

```
if L.size() ≤ 1 then ...... +2 ..... +2
    (L1, L2) ← partition(L, n/2) ..... cn
    recSort(L1) .......... T(n/2)
    recSort(L2) .......... T(n/2)
    L ← merge(L1, L2) ..... cn
return L ........... +1 ..... +1
```

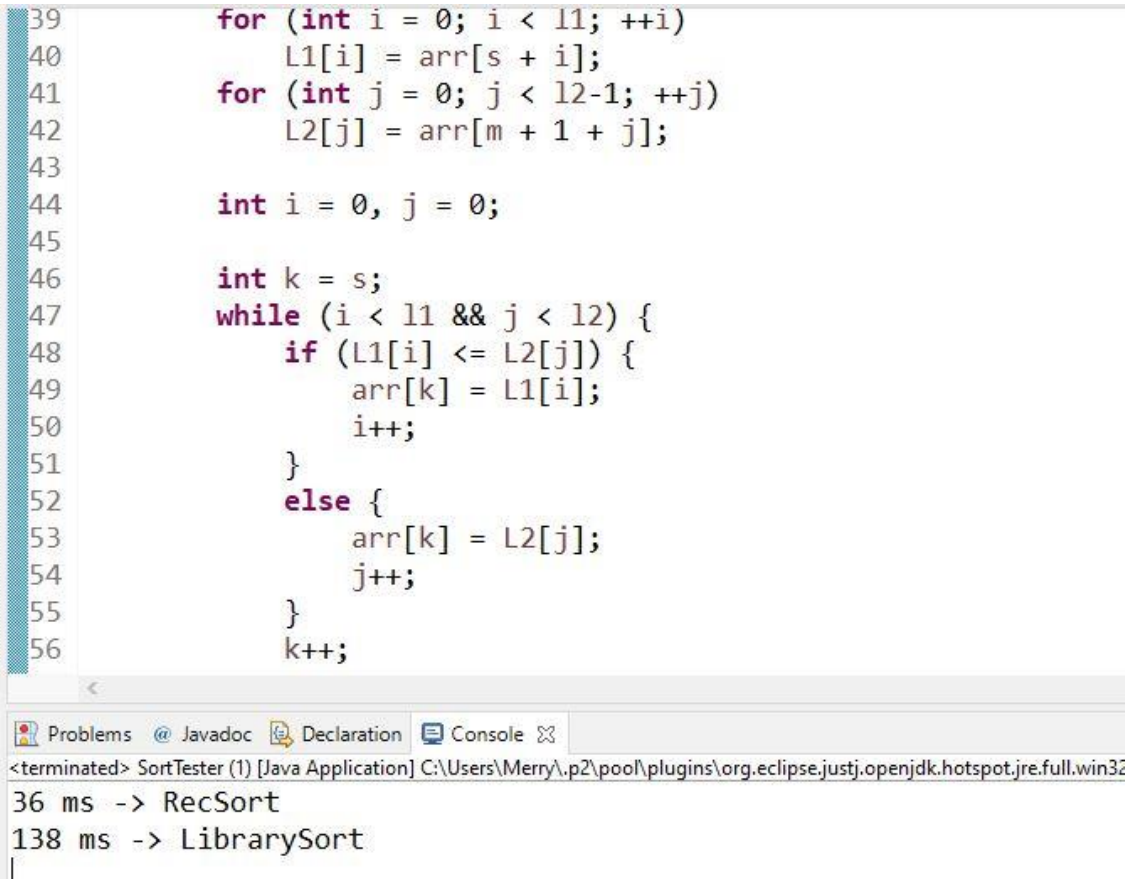
$T(n) = 2(T(n/2) + cn)] + c$

$T(n) = \begin{cases} T(1) = 3 & n=1 \\ T(n) = 2T(n/2) + 2n + c & \text{Otherwise} \end{cases}$

$T(n) \begin{cases} 3 \\ 2T(n/2) + cn + c. \end{cases}$

$a = 2$ $c = c$ $k = 1$
$b = 2$ $d = 3$
$a = b^k$

from the formula $O(n \log n)$



```java
        for (int i = 0; i < l1; ++i)
            L1[i] = arr[s + i];
        for (int j = 0; j < l2-1; ++j)
            L2[j] = arr[m + 1 + j];

        int i = 0, j = 0;

        int k = s;
        while (i < l1 && j < l2) {
            if (L1[i] <= L2[j]) {
                arr[k] = L1[i];
                i++;
            }
            else {
                arr[k] = L2[j];
                j++;
            }
            k++;
```

```
<terminated> SortTester (1) [Java Application] C:\Users\Merry\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
36 ms -> RecSort
138 ms -> LibrarySort
```

a