Algorithm – CS435 / Lab1

Enkhjargal Gansukh – 611031
Erdenesaikhan Tserendagva - 611104
Erkhbayar Ganzorig - 611011

1. **Determine the asymptotic running time of the following procedure (an exact computation of number of basic operations is not necessary):**
   Runs in O(n) + O(n^2) time. This is O(n^2)

2. **Merge pseudo code**

A. Algorithm mergeSortedArrays(a,b)

```
Input array a,b
Output merged array C
n1 <- a.lenght()
n2 <- b.length()
while i < n1 and j < n2 do                    O(n)
        if a[i] < b[j] then
                c[k++] <- a[i++]
        else
                c[k++] <- b[j++]
while i < n1 do                               O(n)
        c[k++] <- a[i++]
while j< n2 do                                O(n)
        c[k++] <- b[j++]
return c
```

B. Analysis

T(n) is 3*O(n) = O(n)
Therefore, the running time of merge sorted arrays algorithm is O(n).

C.
```
public static int[] merge(int[] A, int[] B) {
    int n = A.length;
    int m = B.length;
    int k = 0, i = 0, j = 0;
    int[] C = new int[n + m];
    while (i < n && j < m) {
        if (A[i] < B[j]) {
            C[k] = A[i];
            i++;
        } else {
            C[k] = B[j];
            j++;
        }
        k++;
    }
    while (i < n) {
```

```
            C[k] = A[i];
            i++;
            k++;
        }
        while (j < m) {
            C[k] = B[j];
            j++;
            k++;
        }
        return C;
    }

    public static void main(String[] args) {
        int[] A = {1, 4, 5, 8, 17};
        int[] B = {2, 4, 8, 11, 13, 21, 23, 25};
        System.out.println(Arrays.toString(merge(A, B)));
    }
```

3. Big-oh and Little-oh.

A. $1 + 4n^2$ is $O(n^2)$

$$\lim_{n\to\infty}\left(\frac{1+4n^2}{n^2}\right) = \lim_{n\to\infty}\left(\frac{\frac{1}{n^2}+\frac{4n^2}{n^2}}{\frac{n^2}{n^2}}\right) = 0 + 4 = 4$$

B. $n^2 - 2n$ is not $O(n)$

$$\lim_{n\to\infty}\left(\frac{n^2-2n}{n}\right) = \lim_{n\to\infty}\left(\frac{n(n-2)}{n}\right) = \lim_{n\to\infty}(n-2) = \infty - 2 = \infty$$

C. $\log(n)$ is $o(n)$

$$\lim_{n\to\infty}\left(\frac{\log n}{n}\right) = \lim_{n\to\infty}\left(\frac{\log n'}{n'}\right) = \lim_{n\to\infty}\left(\frac{1}{n\ ln2}\right) = \frac{1}{\infty} = 0$$

D. n is not $o(n)$

$$\lim_{n\to\infty}\left(\frac{n}{n}\right) = 1$$

4. Power Set Algorithm.

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class PowerSet {
    public static void main(String[] args) {
        List<Integer> X = new ArrayList<Integer>(Arrays.asList(new Integer[] {1, 3, 7,
                5, 4, 0, 7, 5}));

        for (int i = 0; i < X.size(); i++) {
            System.out.println(X.get(i));
        }
    }

    public static List<Set<Integer>> set(List<Integer> x) {
        // P <- new list
        List<Set<Integer>> p = new ArrayList<Set<Integer>>();
        // S <- new empty Set
        Set<Integer> s = new HashSet<Integer>();
        p.add(s);
        // T <- new set
        Set<Integer> t;
        while (!x.isEmpty()) {
            int f = x.remove(0);

            for (Set<Integer> set : p) {
                t = new HashSet<Integer>();
                t.add(f);
                t.addAll(set);
                p.add(t);
            }
        }
        return null;
    }
}
```