**Problem1:**

A piece of paper with writing

Description automatically generated with medium confidence

**Problem2: (A)**

**Algorithm:** merge(arr1,arr2)

**Input:** Two already sorted arrays i.e., arr1 and arr2.

**Output:** merged sorted array i.e., resultArray.

**Initial values:** i =0, j =0 , k = 0

While i < arr1.length && j < arr2.length , do

If arr1[i] < arr2[j]

resultArray[k] = arr1[i]

increase i

increase k

else

resultArray[k] = arr2[j]

increase j

increase k

While i < arr1.length

resultArray[k] = arr1[i]

increase i

increase k

While j < arr2.length

resultArray[k] = arr2[j]

increase j

increase k

return resultArray

**Problem2: (B)**

After examining I found that the asymptotic running time is **O(n + m),** where n is length of first array and m is length of second array.

**Problem2: (C)**

**public** **static** **int**[] mergeArray(**int**[] arr1, **int**[] arr2) {

**int**[] resultArray = **new** **int**[arr1.length+arr2.length];

**int** i = 0;

**int** j = 0;

**int** k = 0;

**do** {

**if**(arr1[i] < arr2[j]) {

resultArray[k] = arr1[i];

i++;

}**else** {

resultArray[k] = arr2[j];

j++;

}

k++;

}**while**(i < arr1.length && j < arr2.length);

**while**(i < arr1.length) {

resultArray[k] = arr1[i];

i++;

k++;

}

**while**(j < arr2.length) {

resultArray[k] = arr2[j];

j++;

k++;

}

**return** resultArray;

}

**Problem3: Asymptotic running time of recursive factorial**

Calls : n-1, n-2, n-3, …, 1

Total call : n-1

Running time T(n) = o(n)

**Problem4: Power Set Algorithm**

**public** **static** List<Set<Integer>> powerSet(List<Integer> x) {

List<Set<Integer>> p = **new** ArrayList<>();

Set<Integer> s = **new** HashSet<>();

p.add(s);

Set<Integer> t = **new** HashSet<>();

Iterator<Integer> iterator = x.iterator();

**while**(iterator.hasNext()){

Integer f = iterator.next();

**for**(**int** i = 0 ; i < p.size(); i++){

t.add(f);

p.add(t);

}

}

**return** p;

}

**Problem5: Asymptotic running time using the Master Formula**

We have, T(n) = T(n/2) + n

Where, a = 1, b = 2, c = 1 and k = 1

Since a < bk, we conclude that T(n) is Θ (n).