CSE 222 Homework 4 Report

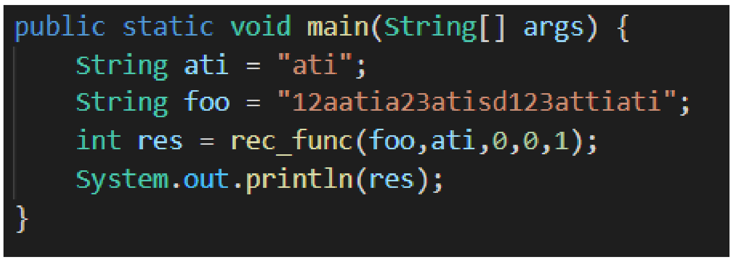
Atakan AKDOGAN

1801042612

Class Designs and Time Complexities

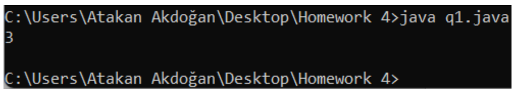
Class Designs:

Q1-)



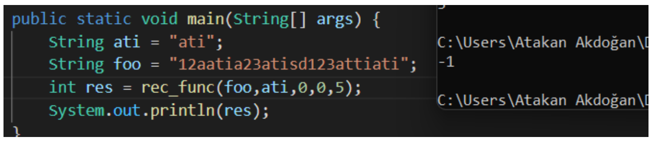
```java
public static void main(String[] args) {
    String ati = "ati";
    String foo = "12aatia23atisd123attiati";
    int res = rec_func(foo,ati,0,0,1);
    System.out.println(res);
}
```

Main method. I created foo String and tried to find hidden little “ati” string in foo. In this example I tried to find 1st ocurrence, and result from this is:



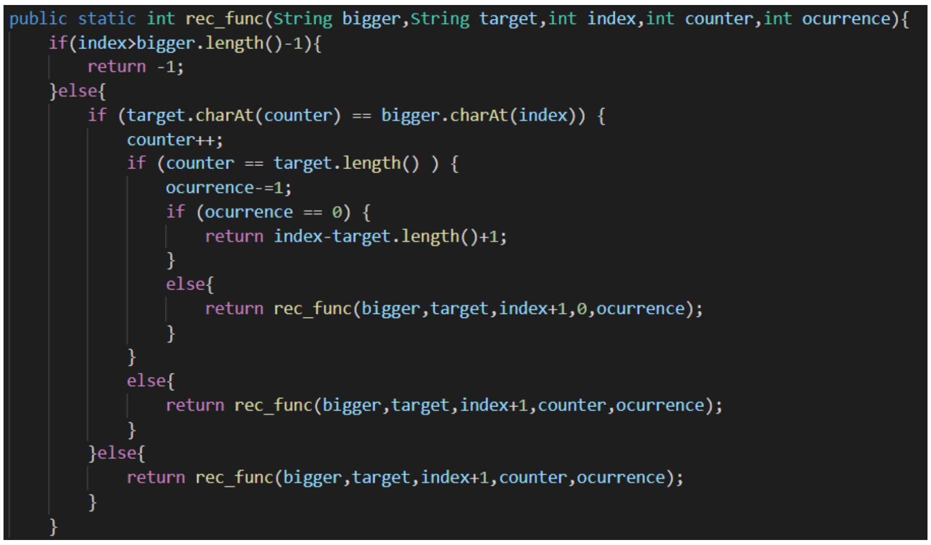
```
C:\Users\Atakan Akdoğan\Desktop\Homework 4>java q1.java
3

C:\Users\Atakan Akdoğan\Desktop\Homework 4>
```

When I try to find 5th ocurrence which is not exist, method returns -1



```java
public static void main(String[] args) {
    String ati = "ati";
    String foo = "12aatia23atisd123attiati";
    int res = rec_func(foo,ati,0,0,5);
    System.out.println(res);
}
```

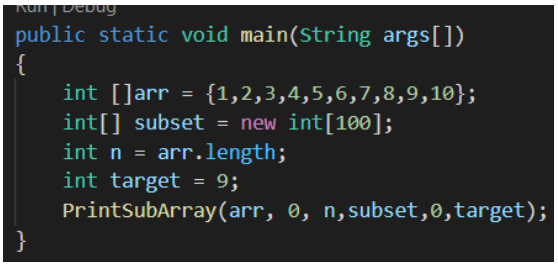
```
C:\Users\Atakan Akdoğan\
-1

C:\Users\Atakan Akdoğan\
```



```java
public static int rec_func(String bigger,String target,int index,int counter,int ocurrence){
    if(index>bigger.length()-1){
        return -1;
    }else{
        if (target.charAt(counter) == bigger.charAt(index)) {
            counter++;
            if (counter == target.length() ) {
                ocurrence-=1;
                if (ocurrence == 0) {
                    return index-target.length()+1;
                }
                else{
                    return rec_func(bigger,target,index+1,0,ocurrence);
                }
            }
            else{
                return rec_func(bigger,target,index+1,counter,ocurrence);
            }
        }else{
            return rec_func(bigger,target,index+1,counter,ocurrence);
        }
    }
```

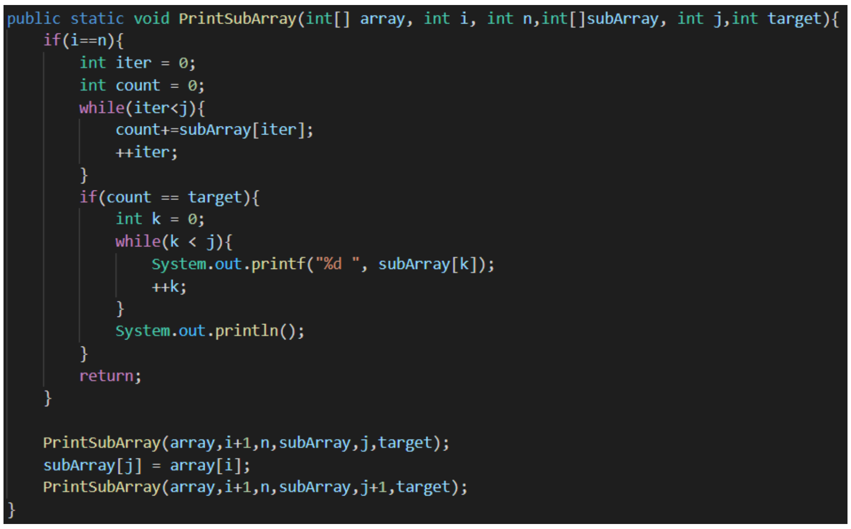
Recursive method part. I approached this problem to compare char by char. If characters equals I count them when method reaches length of target string, ocurrence parameter downs 1 and when it comes 0 it returns index. If doesnt found the string, counter starts from zero.

Q2-)



```java
Run | Debug
public static void main(String args[])
{
    int []arr = {1,2,3,4,5,6,7,8,9,10};
    int[] subset = new int[100];
    int n = arr.length;
    int target = 9;
    PrintSubArray(arr, 0, n,subset,0,target);
}
```

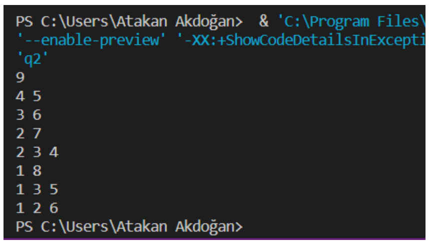
In main method of Q2 class, I choose 1 to 10 array but it can be unsorted doesnt matter. I sent method my array,result array,length of my array and target value. It is void method and it prints the results in recursive method.



```java
public static void PrintSubArray(int[] array, int i, int n,int[]subArray, int j,int target){
    if(i==n){
        int iter = 0;
        int count = 0;
        while(iter<j){
            count+=subArray[iter];
            ++iter;
        }
        if(count == target){
            int k = 0;
            while(k < j){
                System.out.printf("%d ", subArray[k]);
                ++k;
            }
            System.out.println();
        }
        return;
    }

    PrintSubArray(array,i+1,n,subArray,j,target);
    subArray[j] = array[i];
    PrintSubArray(array,i+1,n,subArray,j+1,target);
}
```

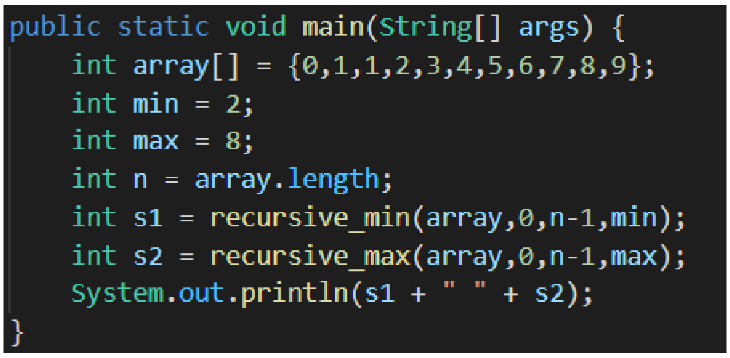
In while I increment count by index of array till subarray ends,when subarray finishes, if the total of indexes is equal to our target, I printed in second while. And sent method 2 times due to more little sizes for subarrays.

Outputs for target 9:



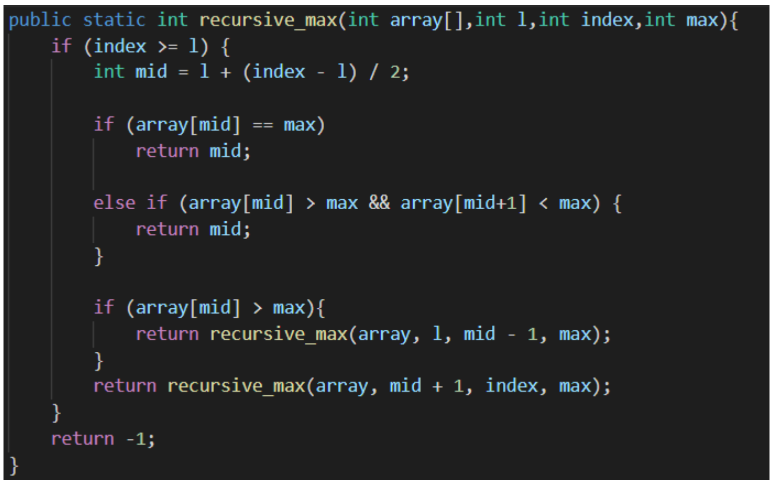
```
PS C:\Users\Atakan Akdoğan>  & 'C:\Program Files\
'--enable-preview' '-XX:+ShowCodeDetailsInExcepti
'q2'
9
4 5
3 6
2 7
2 3 4
1 8
1 3 5
1 2 6
PS C:\Users\Atakan Akdoğan>
```

Q3-)



```java
public static void main(String[] args) {
    int array[] = {0,1,1,2,3,4,5,6,7,8,9};
    int min = 2;
    int max = 8;
    int n = array.length;
    int s1 = recursive_min(array,0,n-1,min);
    int s2 = recursive_max(array,0,n-1,max);
    System.out.println(s1 + " " + s2);
}
```

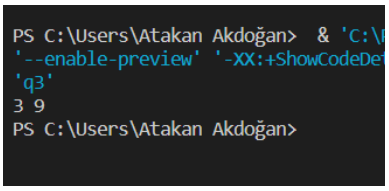
Main method. I created sorted array for binary search. And I choose minimum 2 and maximum 8. I also created 2 rec method for finding minimum and maximum and finally printed indexes.



```java
public static int recursive_max(int array[],int l,int index,int max){
    if (index >= l) {
        int mid = l + (index - l) / 2;

        if (array[mid] == max)
            return mid;

        else if (array[mid] > max && array[mid+1] < max) {
            return mid;
        }

        if (array[mid] > max){
            return recursive_max(array, l, mid - 1, max);
        }
        return recursive_max(array, mid + 1, index, max);
    }
    return -1;
}
```

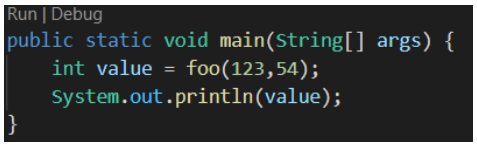
Recursive method for maximum number target finding. Same method for recursive minimum. If our mid is smaller returns mid+1 to recursşve method. If bigger,returns mid-1. If finds,returns.

Output for 2 and 8:

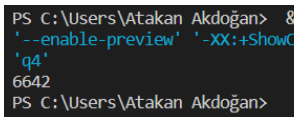


```
PS C:\Users\Atakan Akdoğan>  & 'C:\
'--enable-preview' '-XX:+ShowCodeDe
'q3'
3 9
PS C:\Users\Atakan Akdoğan>
```

Q4-)

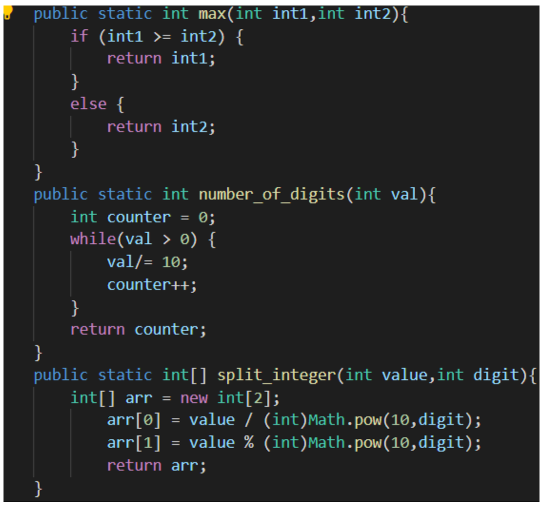


```
Run | Debug
public static void main(String[] args) {
    int value = foo(123,54);
    System.out.println(value);
}
```



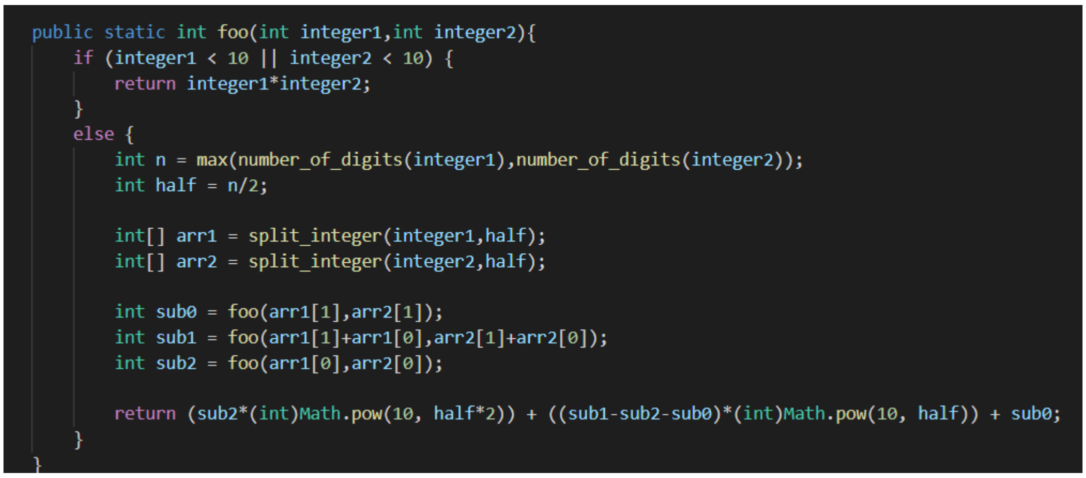
```
PS C:\Users\Atakan Akdoğan>  &
'--enable-preview' '-XX:+ShowC
'q4'
6642
PS C:\Users\Atakan Akdoğan>
```

Main method and output for 123 and 54. It actually does recursive methods and returns multiplication of 2 number.



```
public static int max(int int1,int int2){
    if (int1 >= int2) {
        return int1;
    }
    else {
        return int2;
    }
}
public static int number_of_digits(int val){
    int counter = 0;
    while(val > 0) {
        val/= 10;
        counter++;
    }
    return counter;
}
public static int[] split_integer(int value,int digit){
    int[] arr = new int[2];
        arr[0] = value / (int)Math.pow(10,digit);
        arr[1] = value % (int)Math.pow(10,digit);
        return arr;
}
```

Helpers. Max returns which number is bigger. Number_of_Digits goes till values end and counts number of digit. Split integer is Halfs number from middle(digit) and collects into array for keeping 2 number and returns the array.

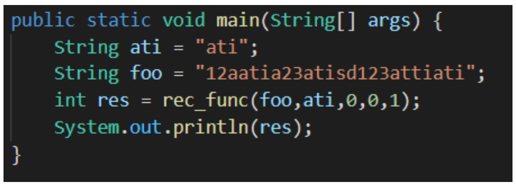


```
public static int foo(int integer1,int integer2){
    if (integer1 < 10 || integer2 < 10) {
        return integer1*integer2;
    }
    else {
        int n = max(number_of_digits(integer1),number_of_digits(integer2));
        int half = n/2;

        int[] arr1 = split_integer(integer1,half);
        int[] arr2 = split_integer(integer2,half);

        int sub0 = foo(arr1[1],arr2[1]);
        int sub1 = foo(arr1[1]+arr1[0],arr2[1]+arr2[0]);
        int sub2 = foo(arr1[0],arr2[0]);

        return (sub2*(int)Math.pow(10, half*2)) + ((sub1-sub2-sub0)*(int)Math.pow(10, half)) + sub0;
    }
}
```

Given foo method.

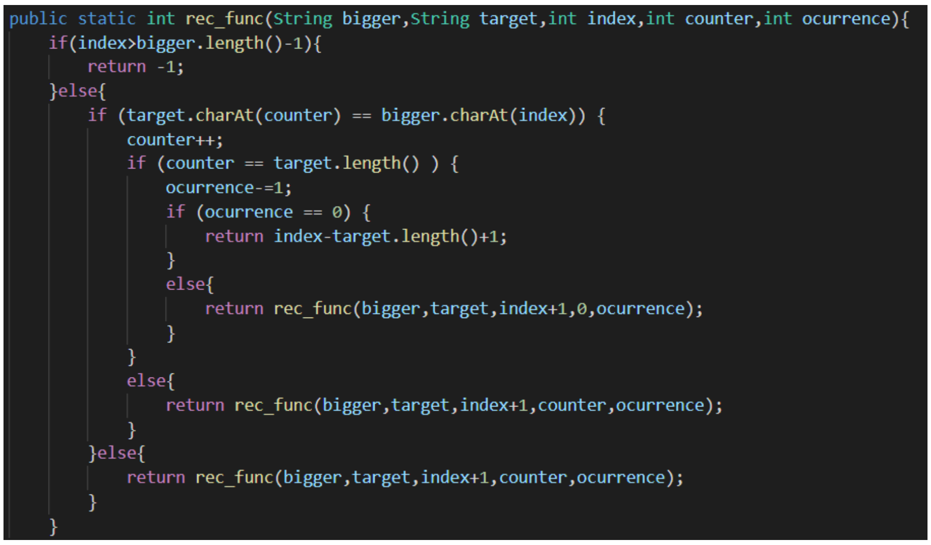
Time Complexities(Explanations in other PDF):

Q1-)



```java
public static void main(String[] args) {
    String ati = "ati";
    String foo = "12aatia23atisd123attiati";
    int res = rec_func(foo,ati,0,0,1);
    System.out.println(res);
}
```

Teta(1)

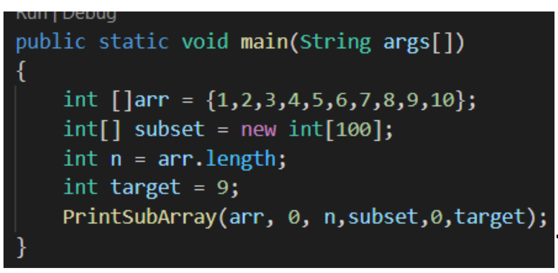


```java
public static int rec_func(String bigger,String target,int index,int counter,int ocurrence){
    if(index>bigger.length()-1){
        return -1;
    }else{
        if (target.charAt(counter) == bigger.charAt(index)) {
            counter++;
            if (counter == target.length() ) {
                ocurrence-=1;
                if (ocurrence == 0) {
                    return index-target.length()+1;
                }
                else{
                    return rec_func(bigger,target,index+1,0,ocurrence);
                }
            }
            else{
                return rec_func(bigger,target,index+1,counter,ocurrence);
            }
        }else{
            return rec_func(bigger,target,index+1,counter,ocurrence);
        }
    }
```

Best Case = Theta(1)

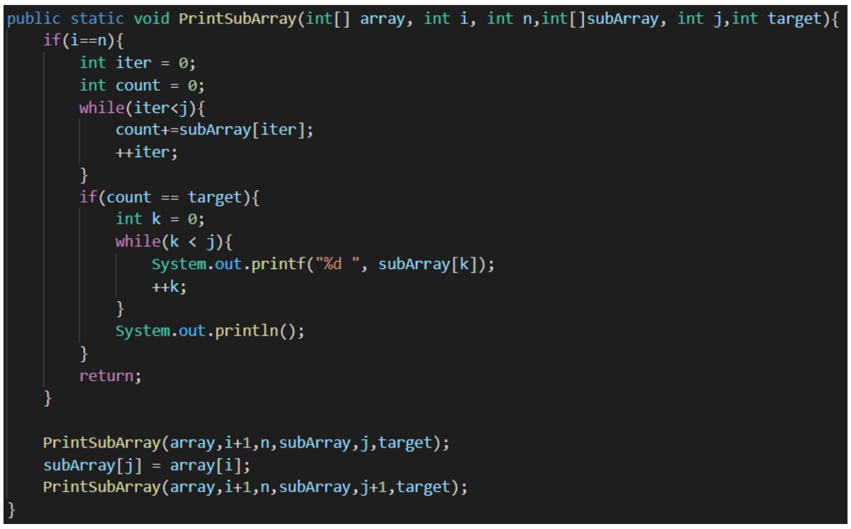
Worst Case = O(n^2)

Q2-)



```java
Run | Debug
public static void main(String args[])
{
    int []arr = {1,2,3,4,5,6,7,8,9,10};
    int[] subset = new int[100];
    int n = arr.length;
    int target = 9;
    PrintSubArray(arr, 0, n,subset,0,target);
}
```

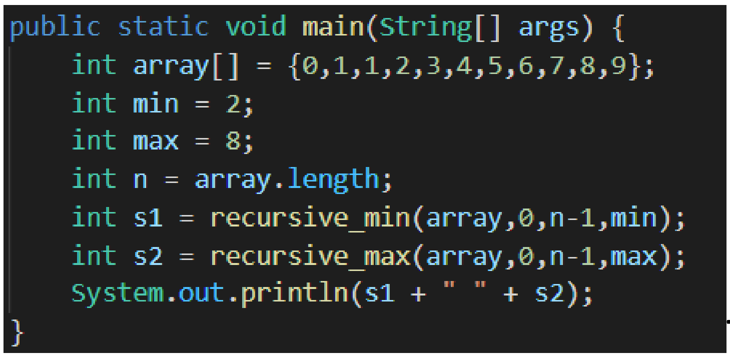
Theta(1)



```
public static void PrintSubArray(int[] array, int i, int n,int[]subArray, int j,int target){
    if(i==n){
        int iter = 0;
        int count = 0;
        while(iter<j){
            count+=subArray[iter];
            ++iter;
        }
        if(count == target){
            int k = 0;
            while(k < j){
                System.out.printf("%d ", subArray[k]);
                ++k;
            }
            System.out.println();
        }
        return;
    }

    PrintSubArray(array,i+1,n,subArray,j,target);
    subArray[j] = array[i];
    PrintSubArray(array,i+1,n,subArray,j+1,target);
}
```

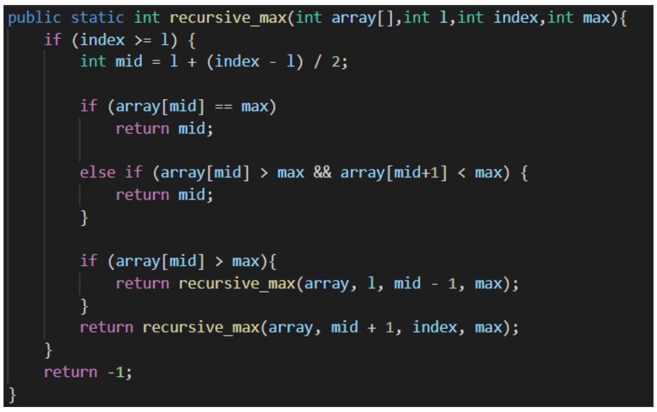
Not has best case. It is Theta(n)

Q3-)



```
public static void main(String[] args) {
    int array[] = {0,1,1,2,3,4,5,6,7,8,9};
    int min = 2;
    int max = 8;
    int n = array.length;
    int s1 = recursive_min(array,0,n-1,min);
    int s2 = recursive_max(array,0,n-1,max);
    System.out.println(s1 + " " + s2);
}
```

Theta(1)

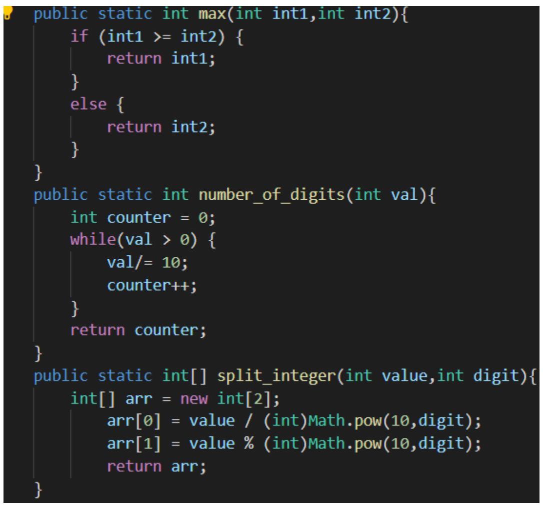


```
public static int recursive_max(int array[],int l,int index,int max){
    if (index >= l) {
        int mid = l + (index - l) / 2;

        if (array[mid] == max)
            return mid;

        else if (array[mid] > max && array[mid+1] < max) {
            return mid;
        }

        if (array[mid] > max){
            return recursive_max(array, l, mid - 1, max);
        }
        return recursive_max(array, mid + 1, index, max);
    }
    return -1;
}
```

Best Case = Theta(1)

Worst Case = O(N^2)

Q4-)

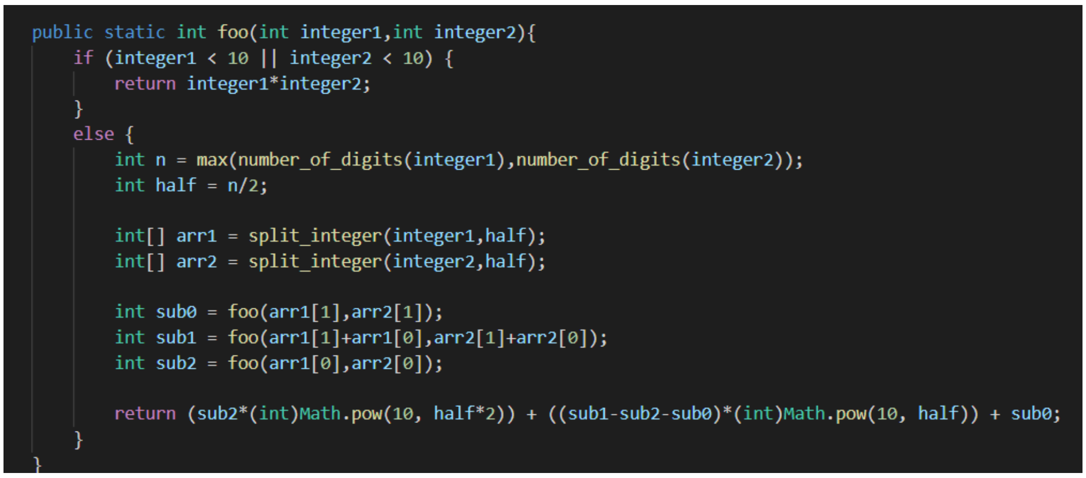


```java
public static int max(int int1,int int2){
    if (int1 >= int2) {
        return int1;
    }
    else {
        return int2;
    }
}
public static int number_of_digits(int val){
    int counter = 0;
    while(val > 0) {
        val/= 10;
        counter++;
    }
    return counter;
}
public static int[] split_integer(int value,int digit){
    int[] arr = new int[2];
        arr[0] = value / (int)Math.pow(10,digit);
        arr[1] = value % (int)Math.pow(10,digit);
        return arr;
}
```

Max -> Theta1

Digits-> Theta(n)

Split->Theta(1)



```java
public static int foo(int integer1,int integer2){
    if (integer1 < 10 || integer2 < 10) {
        return integer1*integer2;
    }
    else {
        int n = max(number_of_digits(integer1),number_of_digits(integer2));
        int half = n/2;

        int[] arr1 = split_integer(integer1,half);
        int[] arr2 = split_integer(integer2,half);

        int sub0 = foo(arr1[1],arr2[1]);
        int sub1 = foo(arr1[1]+arr1[0],arr2[1]+arr2[0]);
        int sub2 = foo(arr1[0],arr2[0]);

        return (sub2*(int)Math.pow(10, half*2)) + ((sub1-sub2-sub0)*(int)Math.pow(10, half)) + sub0;
    }
}
```

Best Case = Theta(1)

Worst Case = O(log base3 n)