GTU Department of Computer Engineering
CSE 222/505 - SPRING 2022
HOMEWORK 4 REPORT

BURAK ÇİÇEK
1901042260

System Requirements

For Q1:
In this question we have 2 methods and 1 ArrayList:
occurenceIndexArrayList - To keep ith occurrences indexes.
public int isContainString(String smallString, String bigString, int indexOfIthOccurence) - For looking occurence of small string in given bigger string takes three parameter returns -1 if indexOfIthOccurence is bigger than existence occurrence number.
private int isContainStringRecur(String smallString, String bigString,int index) - Recursive searcher function starts from index that is 0.

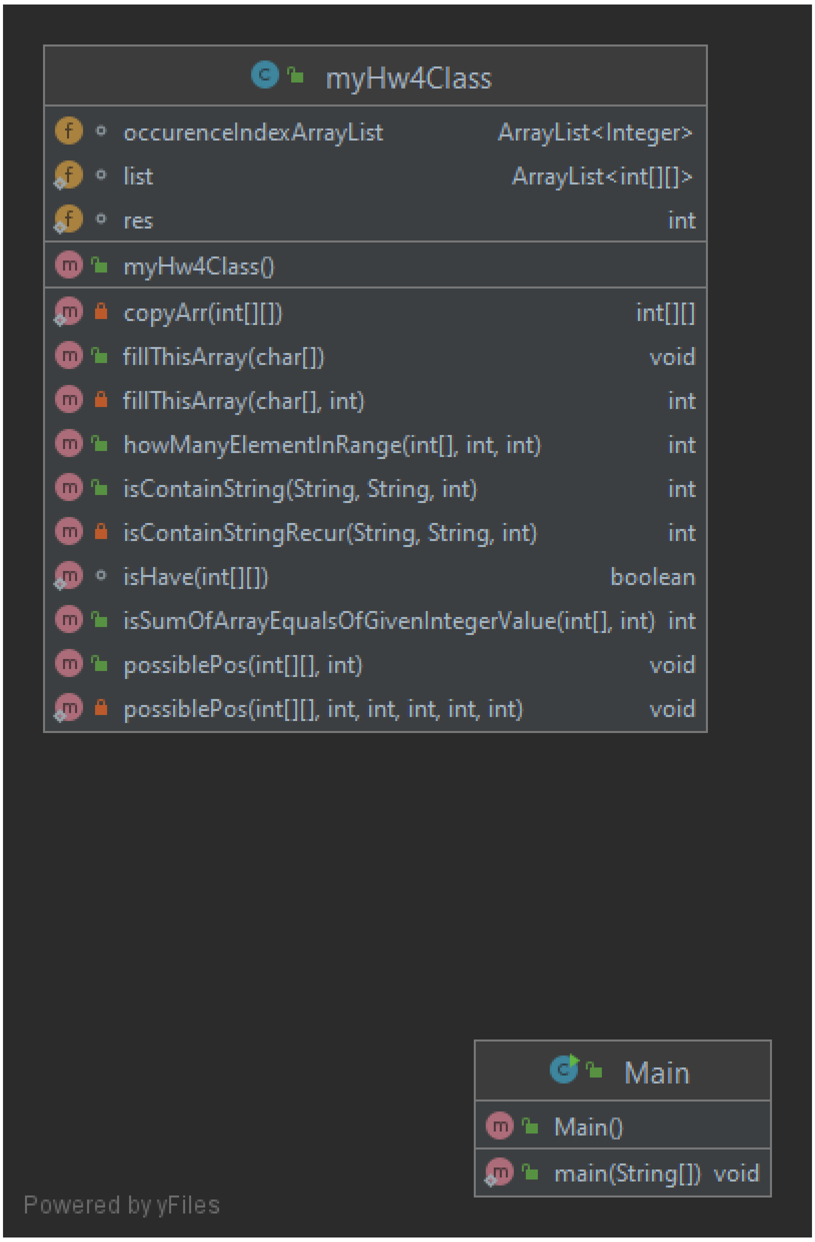
For Q2:
public int howManyElementInRange(int [] givenArray, int lowerBound, int upperBound) - Looking for how many elements in given range.

For Q3:
public int isSumOfArrayEqualsOfGivenIntegerValue(int [] givenArray, int sumValue) - Looking array elements that sums are equal that given value.

For Q5:
public void fillThisArray(char [] givenArray) - To fill array from 3 to L
private int fillThisArray(char [] givenArray,int k) - To fill array starts from 3 to L. This is helper function of Q5.

For Q6: I have 4 methods for this question, one of is for copying array, the other is prevent the duplication and also have main recursive method.
private static boolean isHave(int [][] matrix) - Prevent the duplicated result.
private static int[][] copyArr(int[][] matrix) - For Copying array.
public void possiblePos(int[][] matrix,int snakeLength) - Main snake filler method.
private static void possiblePos(int[][] matrix , int length ,int left,int snakeNum,int i , int j) - Private also recursive helper filler method.

Case Diagrams



myHw4Class
occurenceIndexArrayList ArrayList<Integer>
list ArrayList<int[][]>
res int
myHw4Class()
copyArr(int[][]) int[][]
fillThisArray(char[]) void
fillThisArray(char[], int) int
howManyElementInRange(int[], int, int) int
isContainString(String, String, int) int
isContainStringRecur(String, String, int) int
isHave(int[][]) boolean
isSumOfArrayEqualsOfGivenIntegerValue(int[], int) int
possiblePos(int[][], int) void
possiblePos(int[][], int, int, int, int, int) void
Main
Main()
main(String[]) void
Powered by yFiles

Problem Solution Approach

In fact, the first method I wrote for Question 1 was to stop at the first index of the first substring it found. That's how I understood the question, but when I read it on the homework forum, I can say that when I modified the method by adding an arraylist that holds the indexes and another main method, I got what I wanted in the forum.

I did not use the algorithm suggested for the 2nd question, I designed a better algorithm myself. It starts from the first element and goes up to the lowerbound. After I provide this, I send the same array for the same logical upper bound.

An hint was given for the 3rd question, but I did not set up my recursive algorithm according to this logic. I proceed by adding the elements of the given array each time, and when the sum is equal to the given sum, I print these elements on the screen. In the next recursive call, I delete the first element and copy the array and make a recursive call again.

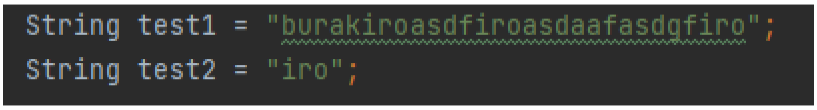
In question 5, my array is always the same, I make a recursive call by increasing the parameter of the block size to be filled. I perform my fillings according to the result of dividing the Array size by the block size. Finally, I can say that I am performing augmented calling.

In the 6th question, I used the backtracking algorithm. I kept the ids of the snakes, and created an algorithm based on the length, fulfilling the relevant condition.

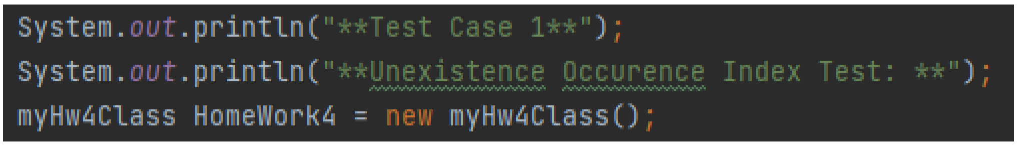
Test Cases

For Q1-

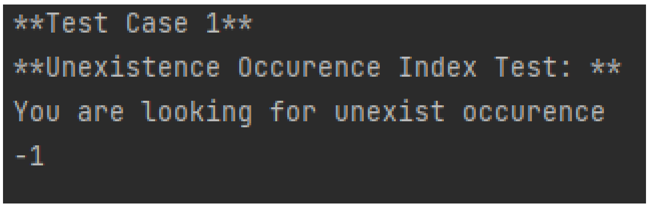
Inputs:



```
String test1 = "burakiroasdfiroasdaafasdgfiro";
String test2 = "iro";
```

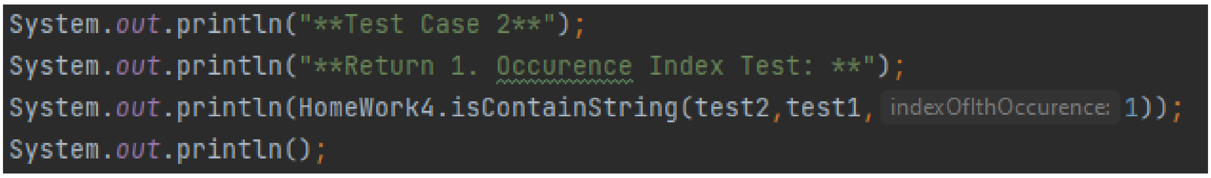


```
System.out.println("**Test Case 1**");
System.out.println("**Unexistence Occurence Index Test: **");
myHw4Class HomeWork4 = new myHw4Class();
```

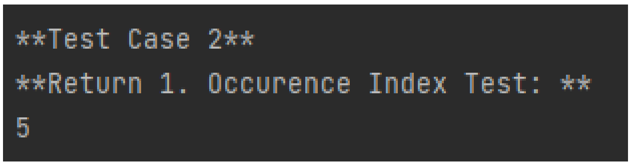


```
**Test Case 1**
**Unexistence Occurence Index Test: **
You are looking for unexist occurence
-1
```

Pass

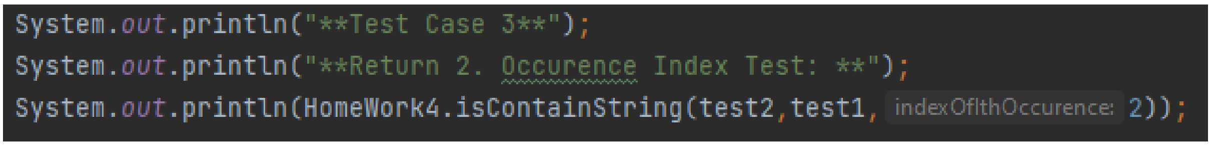


```
System.out.println("**Test Case 2**");
System.out.println("**Return 1. Occurence Index Test: **");
System.out.println(HomeWork4.isContainString(test2,test1, indexOfIthOccurence: 1));
System.out.println();
```

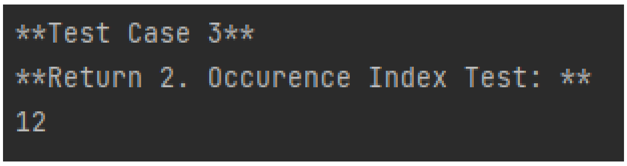


```
**Test Case 2**
**Return 1. Occurence Index Test: **
5
```

Pass

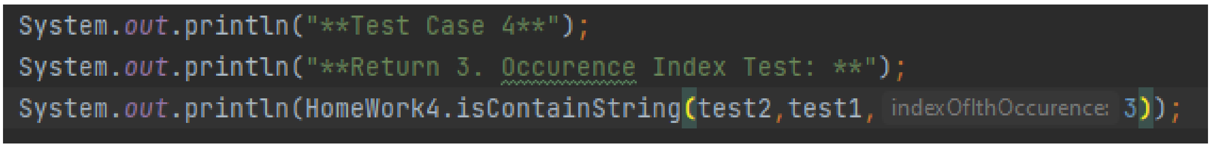


```
System.out.println("**Test Case 3**");
System.out.println("**Return 2. Occurence Index Test: **");
System.out.println(HomeWork4.isContainString(test2,test1, indexOfIthOccurence: 2));
```

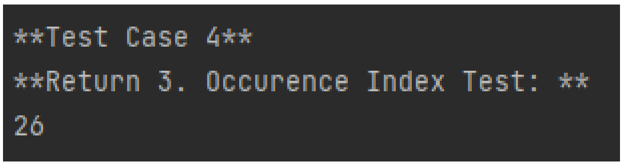


```
**Test Case 3**
**Return 2. Occurence Index Test: **
12
```

Pass

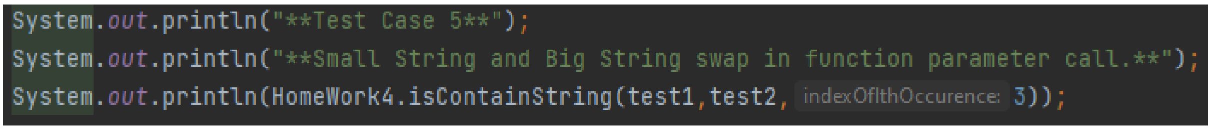


```
System.out.println("**Test Case 4**");
System.out.println("**Return 3. Occurence Index Test: **");
System.out.println(HomeWork4.isContainString(test2,test1, indexOfIthOccurence: 3));
```

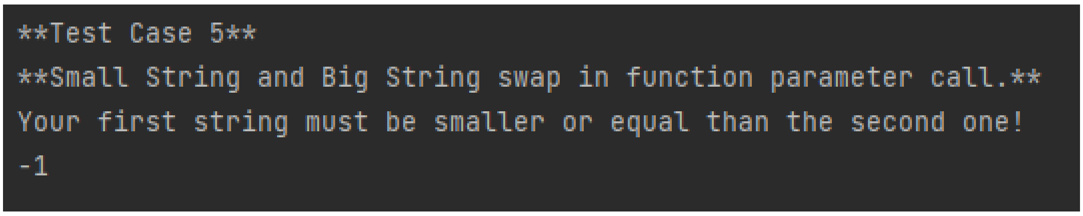


```
**Test Case 4**
**Return 3. Occurence Index Test: **
26
```

Pass



```
System.out.println("**Test Case 5**");
System.out.println("**Small String and Big String swap in function parameter call.**");
System.out.println(HomeWork4.isContainString(test1,test2, indexOfIthOccurence: 3));
```

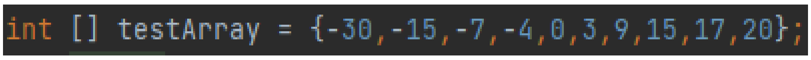


```
**Test Case 5**
**Small String and Big String swap in function parameter call.**
Your first string must be smaller or equal than the second one!
-1
```

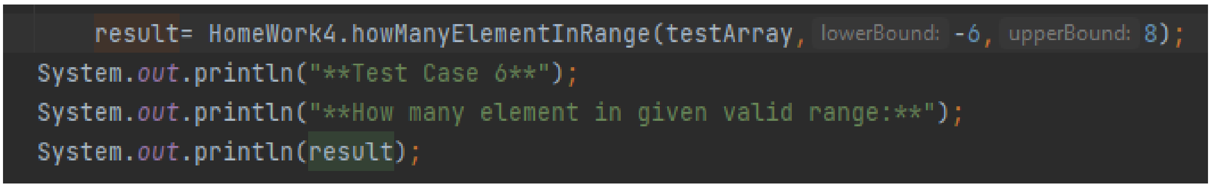
Pass

For Q2-

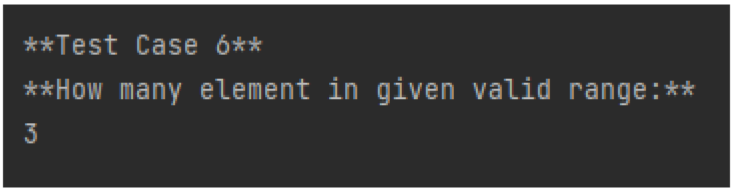
Inputs:



```
int [] testArray = {-30,-15,-7,-4,0,3,9,15,17,20};
```

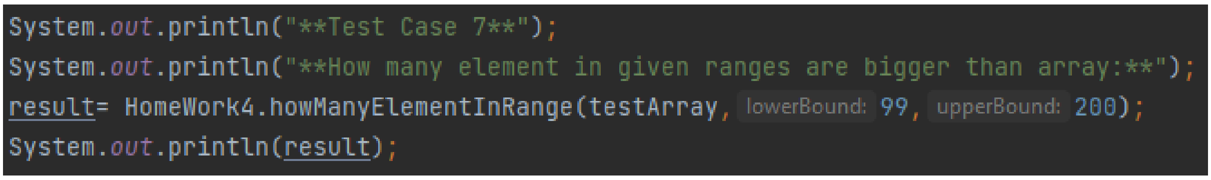


```
    result= HomeWork4.howManyElementInRange(testArray, lowerBound: -6, upperBound: 8);
System.out.println("**Test Case 6**");
System.out.println("**How many element in given valid range:**");
System.out.println(result);
```

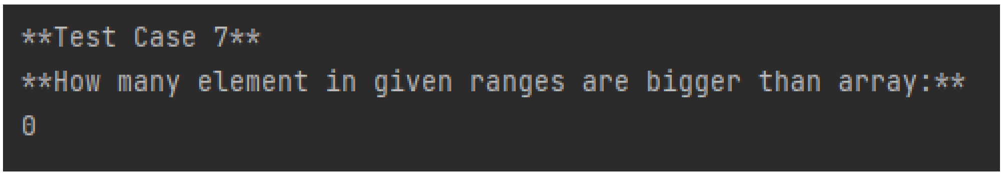


```
**Test Case 6**
**How many element in given valid range:**
3
```

Pass

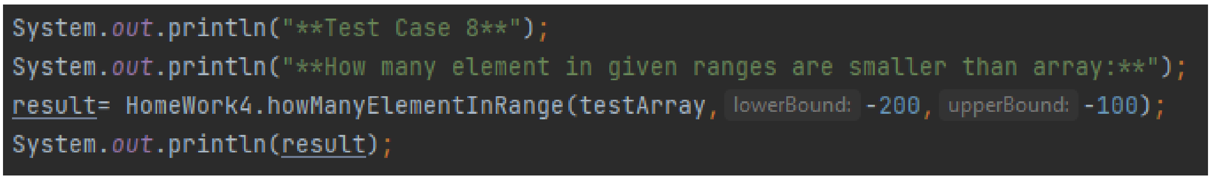


```
System.out.println("**Test Case 7**");
System.out.println("**How many element in given ranges are bigger than array:**");
result= HomeWork4.howManyElementInRange(testArray, lowerBound: 99, upperBound: 200);
System.out.println(result);
```

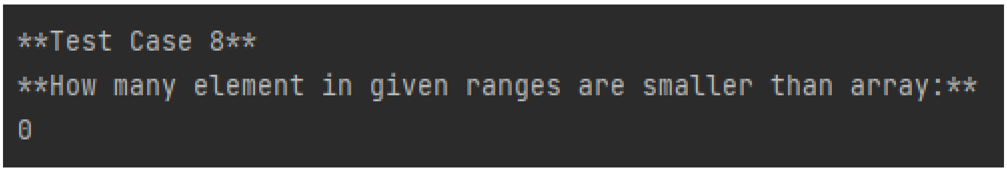


```
**Test Case 7**
**How many element in given ranges are bigger than array:**
0
```

Pass



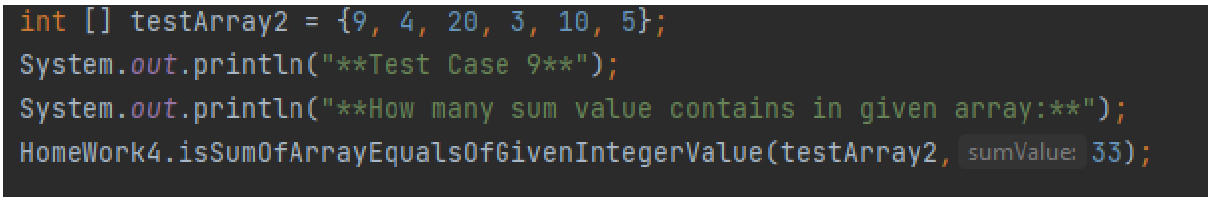
```
System.out.println("**Test Case 8**");
System.out.println("**How many element in given ranges are smaller than array:**");
result= HomeWork4.howManyElementInRange(testArray, lowerBound: -200, upperBound: -100);
System.out.println(result);
```



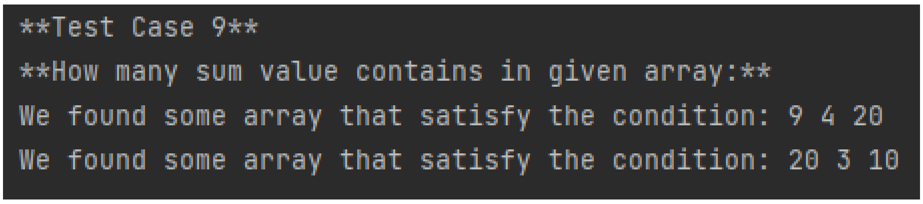
```
**Test Case 8**
**How many element in given ranges are smaller than array:**
0
```

Pass

For Q3-

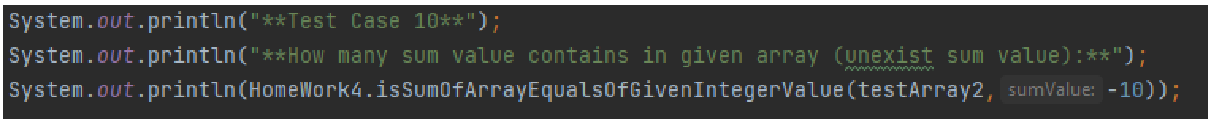


```
int [] testArray2 = {9, 4, 20, 3, 10, 5};
System.out.println("**Test Case 9**");
System.out.println("**How many sum value contains in given array:**");
HomeWork4.isSumOfArrayEqualsOfGivenIntegerValue(testArray2, sumValue: 33);
```

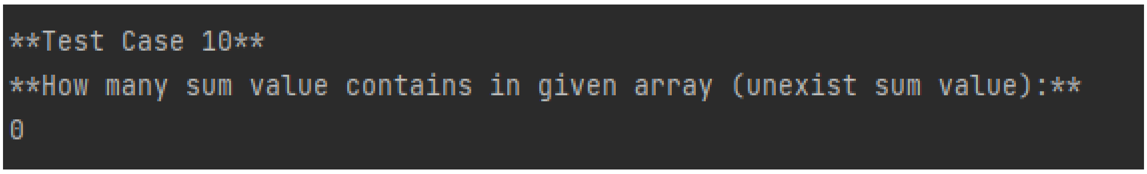


```
**Test Case 9**
**How many sum value contains in given array:**
We found some array that satisfy the condition: 9 4 20
We found some array that satisfy the condition: 20 3 10
```

Pass

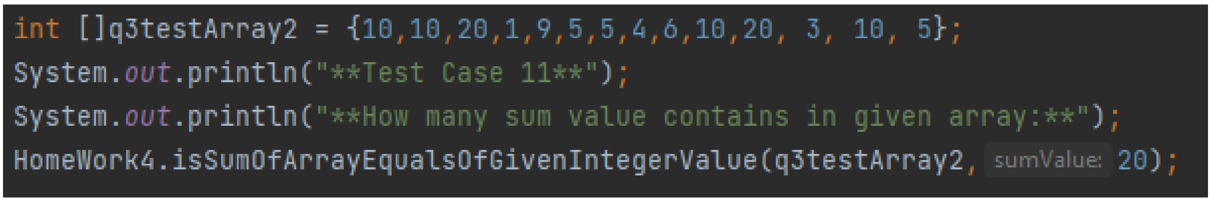


```
System.out.println("**Test Case 10**");
System.out.println("**How many sum value contains in given array (unexist sum value):**");
System.out.println(HomeWork4.isSumOfArrayEqualsOfGivenIntegerValue(testArray2, sumValue: -10));
```

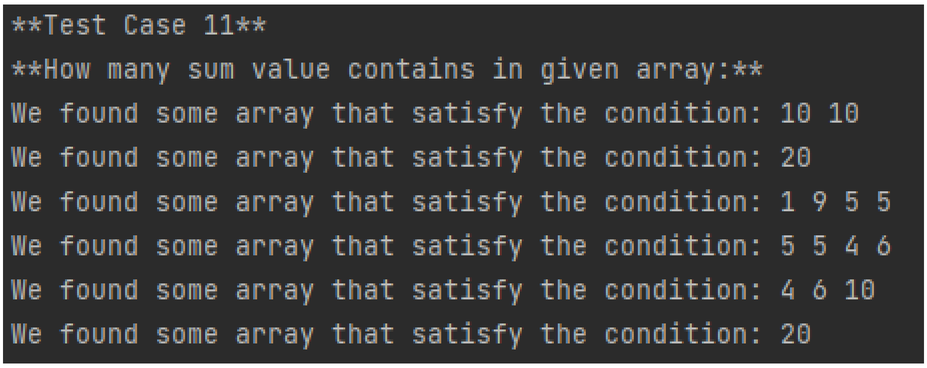


```
**Test Case 10**
**How many sum value contains in given array (unexist sum value):**
0
```

Pass



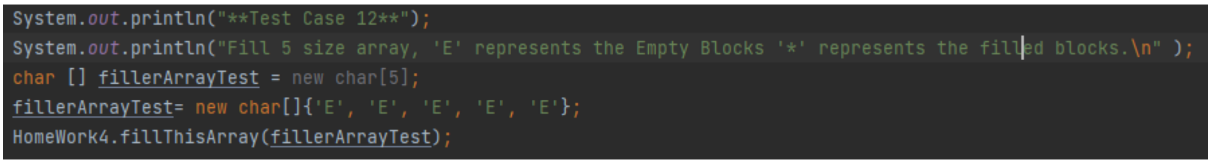
```
int []q3testArray2 = {10,10,20,1,9,5,5,4,6,10,20, 3, 10, 5};
System.out.println("**Test Case 11**");
System.out.println("**How many sum value contains in given array:**");
HomeWork4.isSumOfArrayEqualsOfGivenIntegerValue(q3testArray2, sumValue: 20);
```



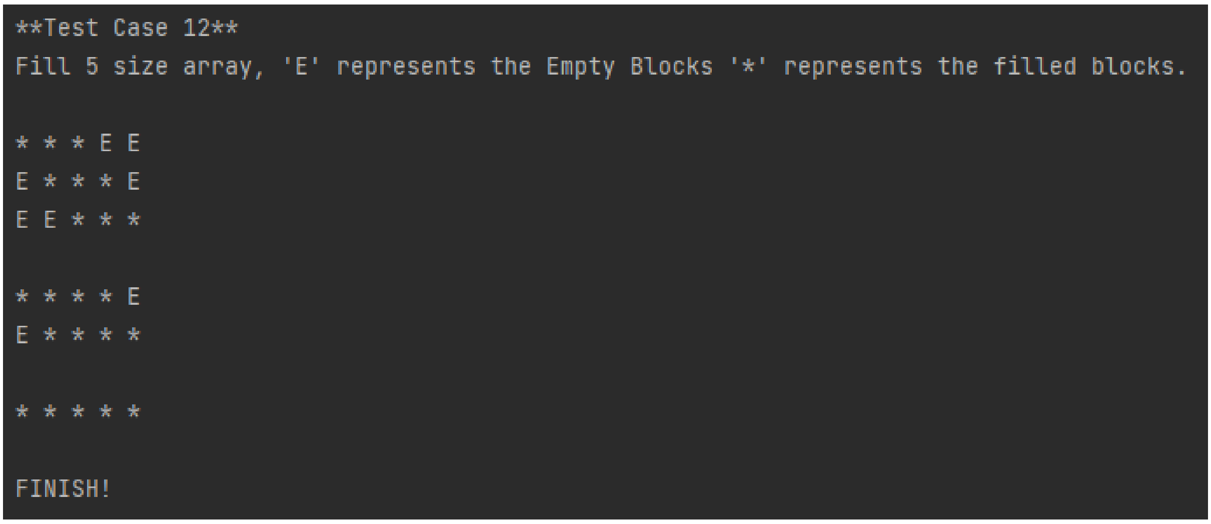
```
**Test Case 11**
**How many sum value contains in given array:**
We found some array that satisfy the condition: 10 10
We found some array that satisfy the condition: 20
We found some array that satisfy the condition: 1 9 5 5
We found some array that satisfy the condition: 5 5 4 6
We found some array that satisfy the condition: 4 6 10
We found some array that satisfy the condition: 20
```

Pass

For Q5-

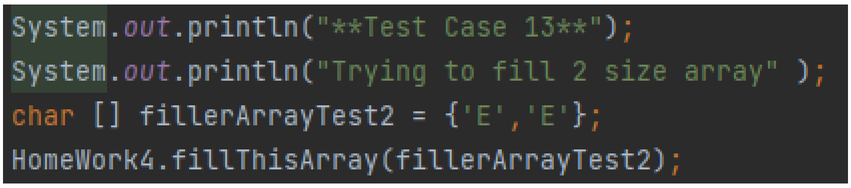


```
System.out.println("**Test Case 12**");
System.out.println("Fill 5 size array, 'E' represents the Empty Blocks '*' represents the filled blocks.\n" );
char [] fillerArrayTest = new char[5];
fillerArrayTest= new char[]{'E', 'E', 'E', 'E', 'E'};
HomeWork4.fillThisArray(fillerArrayTest);
```

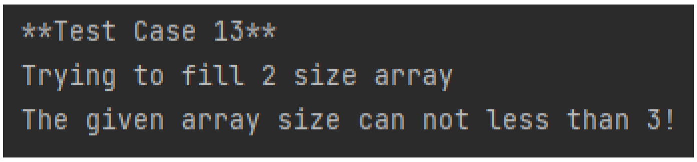


```
**Test Case 12**
Fill 5 size array, 'E' represents the Empty Blocks '*' represents the filled blocks.

* * * E E
E * * * E
E E * * *

* * * * E
E * * * *

* * * * *

FINISH!
```

Pass

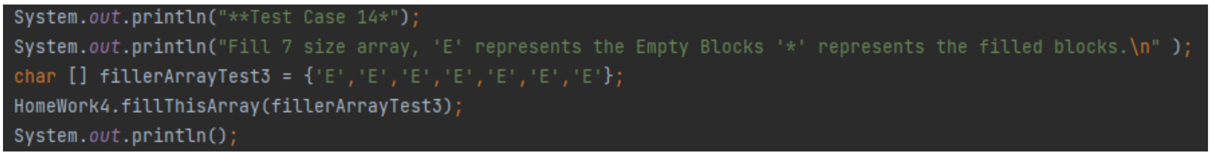


```
System.out.println("**Test Case 13**");
System.out.println("Trying to fill 2 size array" );
char [] fillerArrayTest2 = {'E','E'};
HomeWork4.fillThisArray(fillerArrayTest2);
```

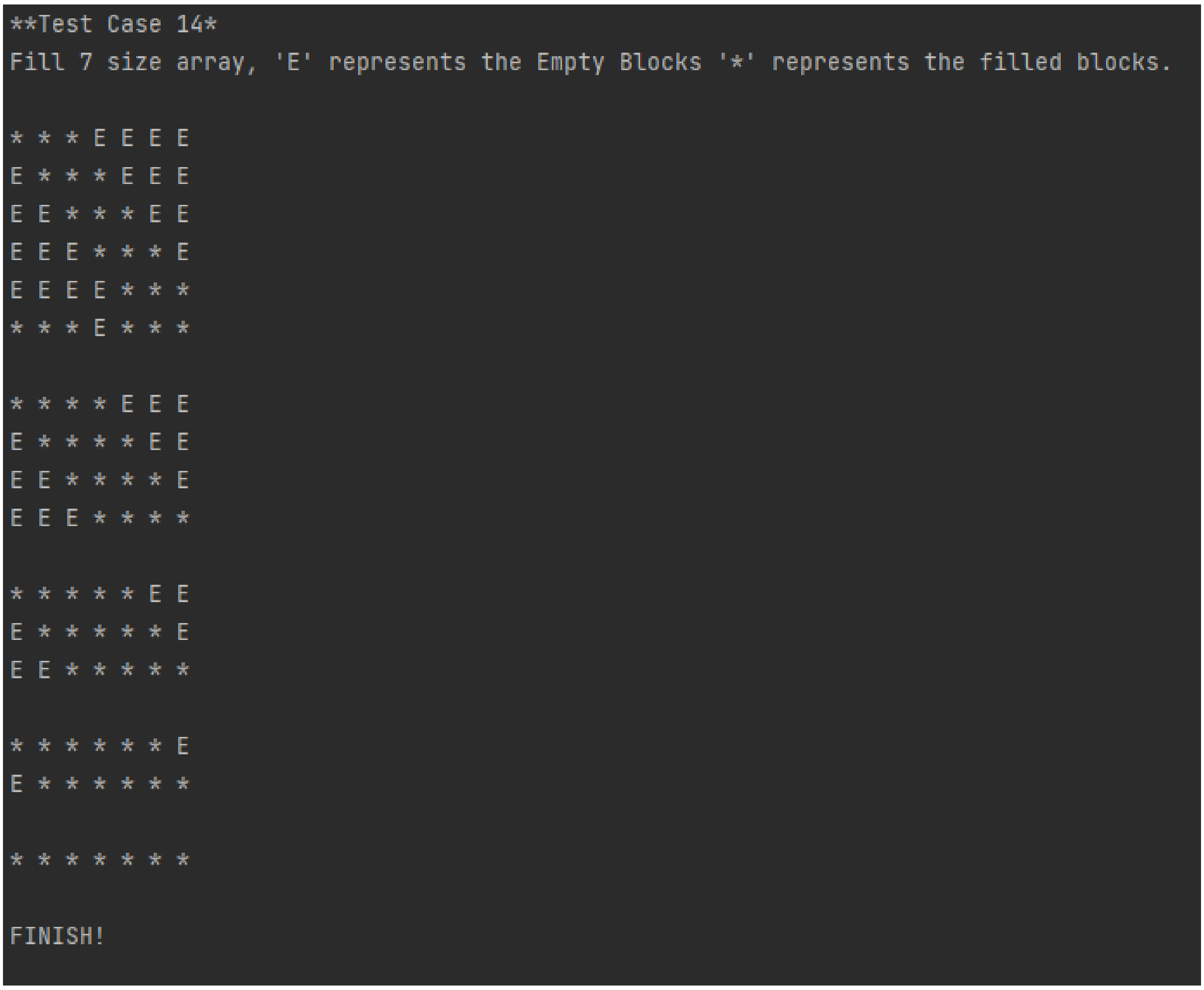


```
**Test Case 13**
Trying to fill 2 size array
The given array size can not less than 3!
```

Pass

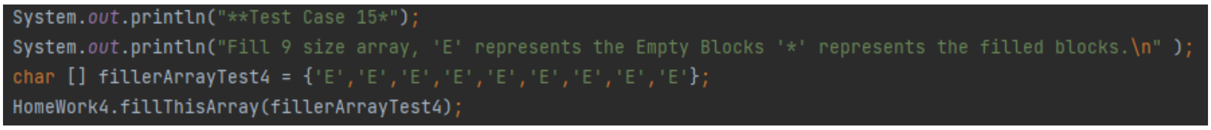


```
System.out.println("**Test Case 14*");
System.out.println("Fill 7 size array, 'E' represents the Empty Blocks '*' represents the filled blocks.\n" );
char [] fillerArrayTest3 = {'E','E','E','E','E','E','E'};
HomeWork4.fillThisArray(fillerArrayTest3);
System.out.println();
```

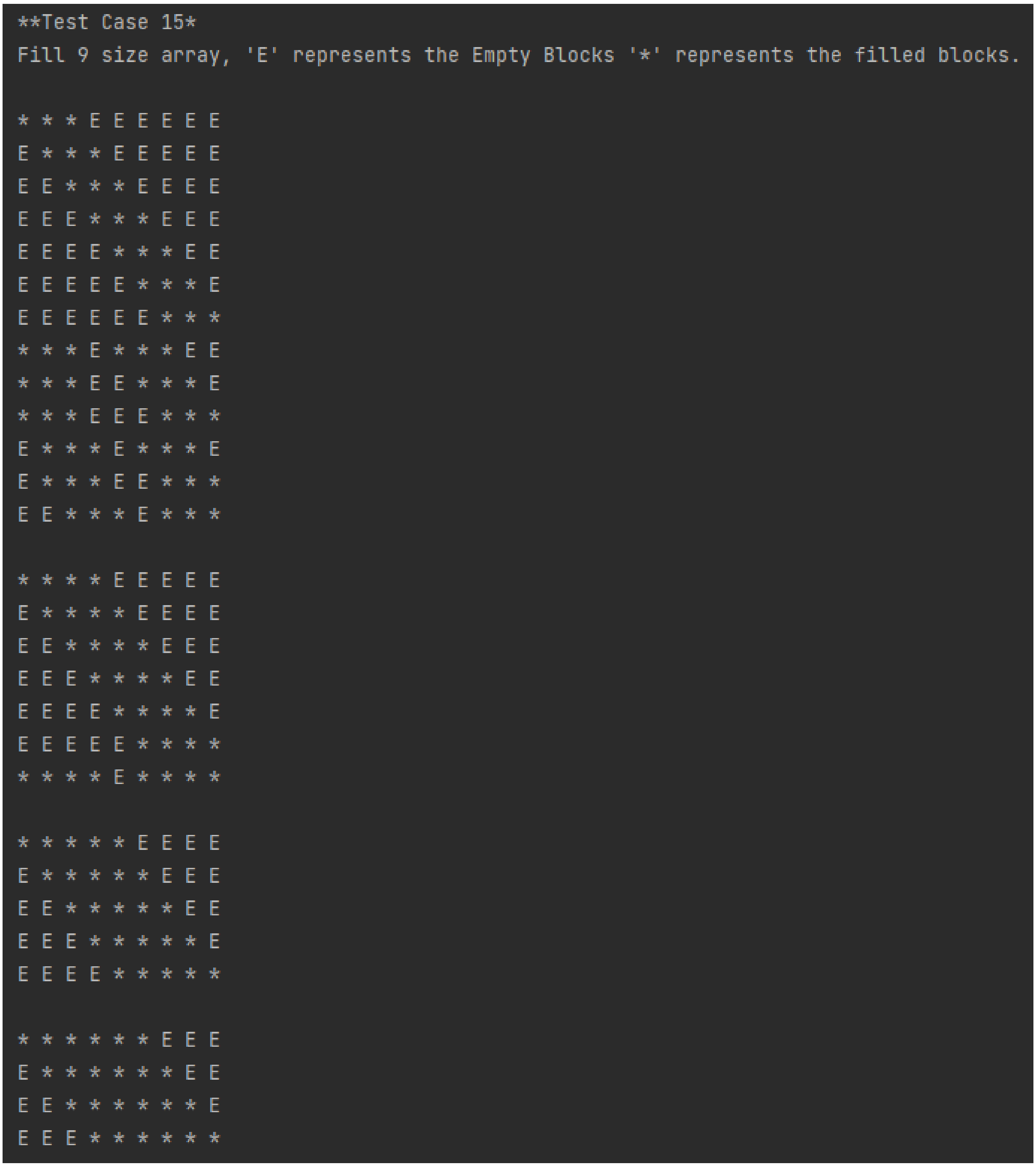


```
**Test Case 14*
Fill 7 size array, 'E' represents the Empty Blocks '*' represents the filled blocks.

* * * E E E E
E * * * E E E
E E * * * E E
E E E * * * E
E E E E * * *
* * * E * * *

* * * * E E E
E * * * * E E
E E * * * * E
E E E * * * *

* * * * * E E
E * * * * * E
E E * * * * *

* * * * * * E
E * * * * * *

* * * * * * *

FINISH!
```

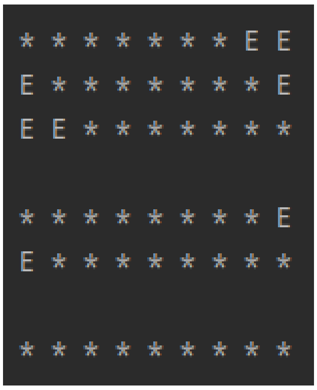
Pass



```
System.out.println("**Test Case 15*");
System.out.println("Fill 9 size array, 'E' represents the Empty Blocks '*' represents the filled blocks.\n" );
char [] fillerArrayTest4 = {'E','E','E','E','E','E','E','E','E'};
HomeWork4.fillThisArray(fillerArrayTest4);
```



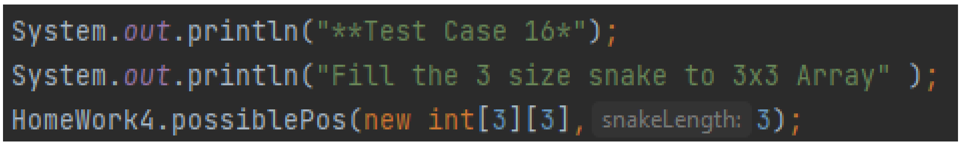
```
**Test Case 15*
Fill 9 size array, 'E' represents the Empty Blocks '*' represents the filled blocks.

* * * E E E E E E
E * * * E E E E E
E E * * * E E E E
E E E * * * E E E
E E E E * * * E E
E E E E E * * * E
E E E E E E * * *
* * * E * * * E E
* * * E E * * * E
* * * E E E * * *
E * * * E * * * E
E * * * E E * * *
E E * * * E * * *

* * * * E E E E E
E * * * * E E E E
E E * * * * E E E
E E E * * * * E E
E E E E * * * * E
E E E E E * * * *
* * * * E * * * *

* * * * * E E E E
E * * * * * E E E
E E * * * * * E E
E E E * * * * * E
E E E E * * * * *

* * * * * * E E E
E * * * * * * E E
E E * * * * * * E
E E E * * * * * *
```



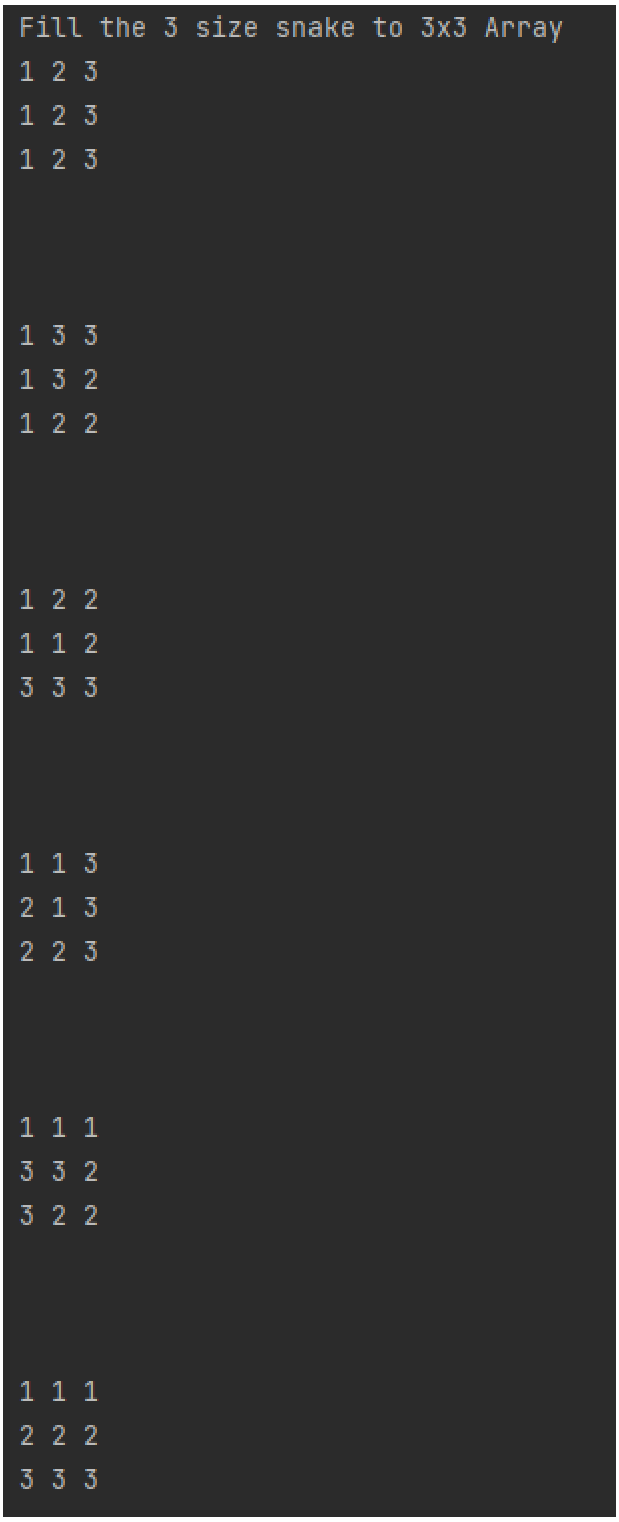
```
* * * * * * * E E
E * * * * * * * E
E E * * * * * * *

* * * * * * * * E
E * * * * * * * *

* * * * * * * * *
```

Pass

For Q6-

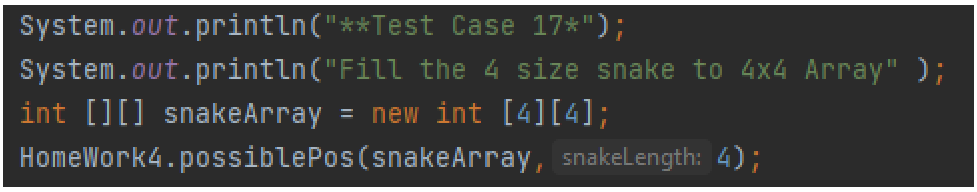


```
System.out.println("**Test Case 16*");
System.out.println("Fill the 3 size snake to 3x3 Array" );
HomeWork4.possiblePos(new int[3][3], snakeLength: 3);
```

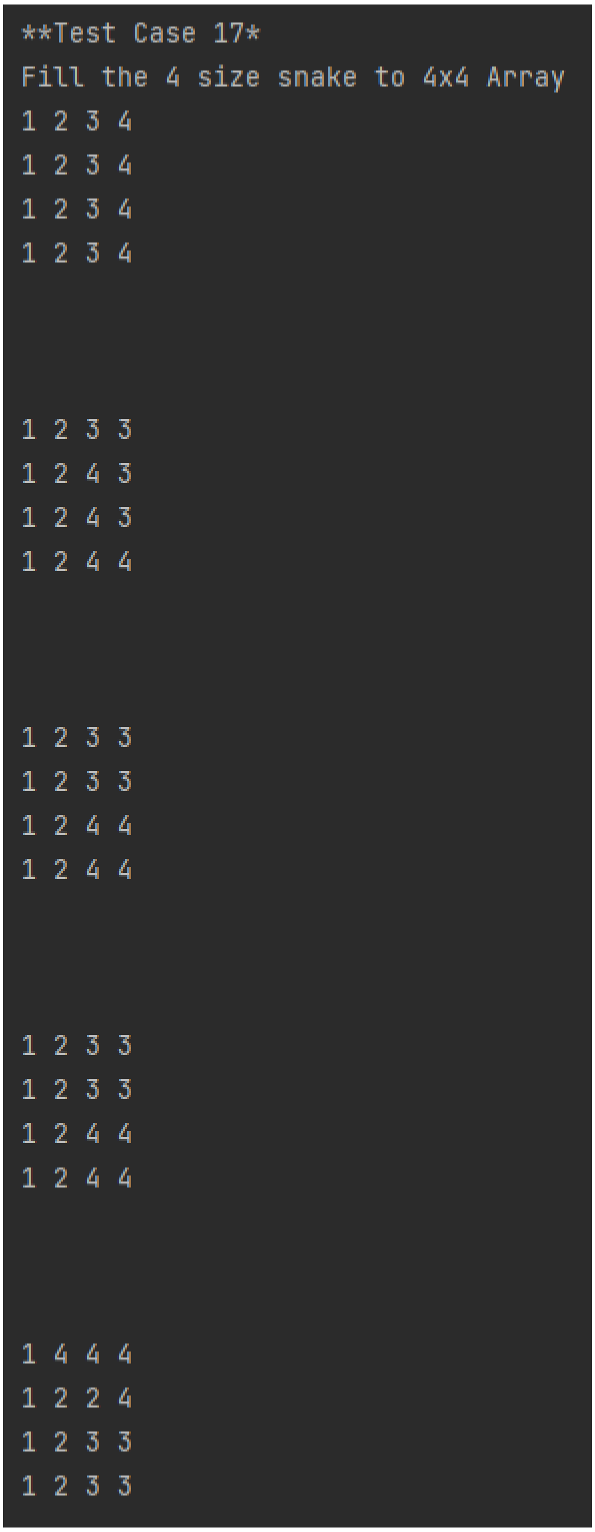


```
Fill the 3 size snake to 3x3 Array
1 2 3
1 2 3
1 2 3



1 3 3
1 3 2
1 2 2



1 2 2
1 1 2
3 3 3



1 1 3
2 1 3
2 2 3



1 1 1
3 3 2
3 2 2



1 1 1
2 2 2
3 3 3
```

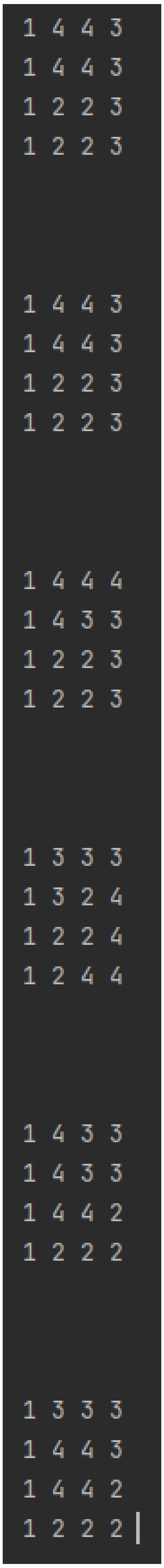
Pass



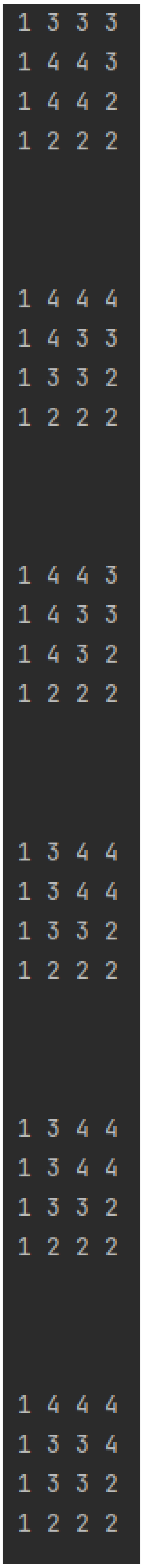
```
System.out.println("**Test Case 17*");
System.out.println("Fill the 4 size snake to 4x4 Array" );
int [][] snakeArray = new int [4][4];
HomeWork4.possiblePos(snakeArray, snakeLength: 4);
```



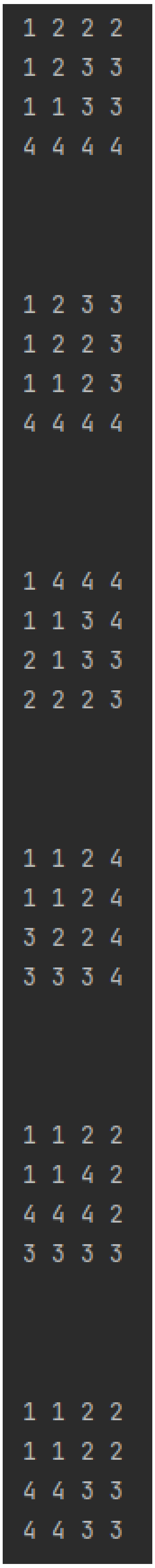
```
**Test Case 17*
Fill the 4 size snake to 4x4 Array
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4



1 2 3 3
1 2 4 3
1 2 4 3
1 2 4 4



1 2 3 3
1 2 3 3
1 2 4 4
1 2 4 4



1 2 3 3
1 2 3 3
1 2 4 4
1 2 4 4



1 4 4 4
1 2 2 4
1 2 3 3
1 2 3 3
```



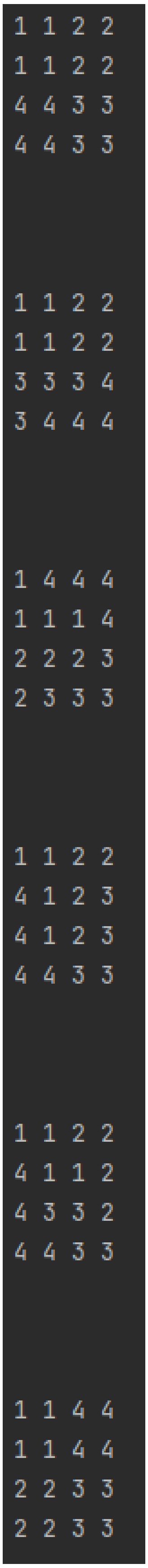
```
1 4 4 3
1 4 4 3
1 2 2 3
1 2 2 3



1 4 4 3
1 4 4 3
1 2 2 3
1 2 2 3



1 4 4 4
1 4 3 3
1 2 2 3
1 2 2 3



1 3 3 3
1 3 2 4
1 2 2 4
1 2 4 4



1 4 3 3
1 4 3 3
1 4 4 2
1 2 2 2



1 3 3 3
1 4 4 3
1 4 4 2
1 2 2 2
```



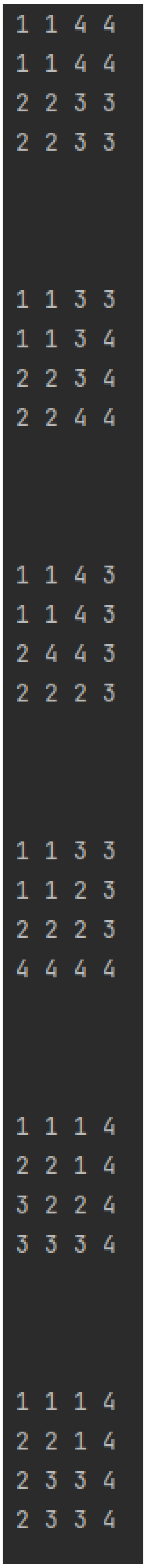
```
1 3 3 3
1 4 4 3
1 4 4 2
1 2 2 2



1 4 4 4
1 4 3 3
1 3 3 2
1 2 2 2



1 4 4 3
1 4 3 3
1 4 3 2
1 2 2 2



1 3 4 4
1 3 4 4
1 3 3 2
1 2 2 2



1 3 4 4
1 3 4 4
1 3 3 2
1 2 2 2



1 4 4 4
1 3 3 4
1 3 3 2
1 2 2 2
```



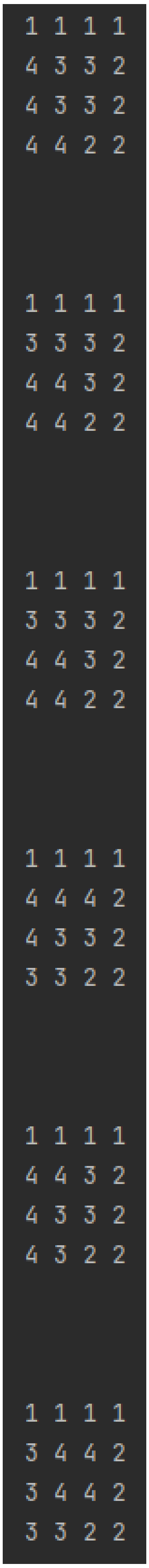
```
1 2 2 2
1 2 3 3
1 1 3 3
4 4 4 4



1 2 3 3
1 2 2 3
1 1 2 3
4 4 4 4



1 4 4 4
1 1 3 4
2 1 3 3
2 2 2 3



1 1 2 4
1 1 2 4
3 2 2 4
3 3 3 4



1 1 2 2
1 1 4 2
4 4 4 2
3 3 3 3



1 1 2 2
1 1 2 2
4 4 3 3
4 4 3 3
```



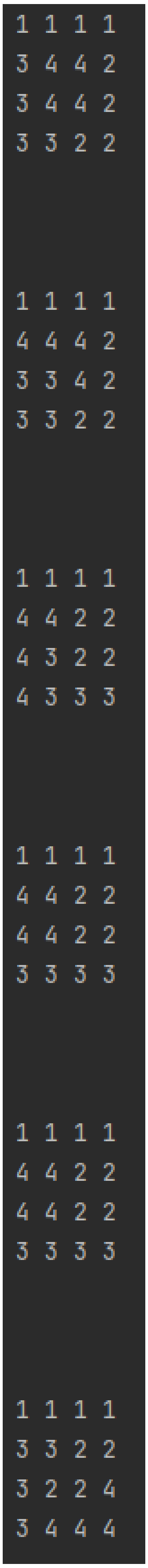
```
1 1 2 2
1 1 2 2
4 4 3 3
4 4 3 3



1 1 2 2
1 1 2 2
3 3 3 4
3 4 4 4



1 4 4 4
1 1 1 4
2 2 2 3
2 3 3 3



1 1 2 2
4 1 2 3
4 1 2 3
4 4 3 3



1 1 2 2
4 1 1 2
4 3 3 2
4 4 3 3



1 1 4 4
1 1 4 4
2 2 3 3
2 2 3 3
```



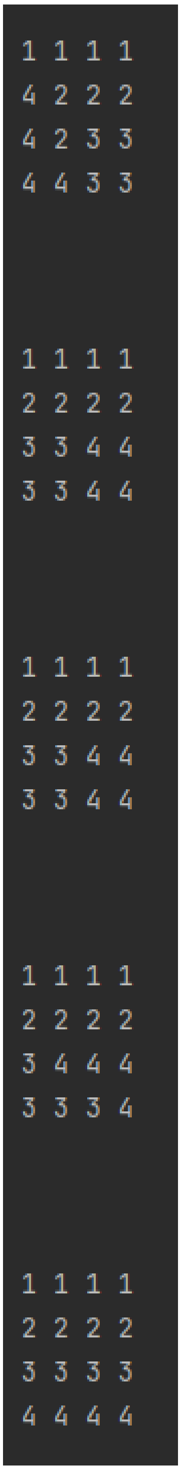
```
1 1 4 4
1 1 4 4
2 2 3 3
2 2 3 3



1 1 3 3
1 1 3 4
2 2 3 4
2 2 4 4



1 1 4 3
1 1 4 3
2 4 4 3
2 2 2 3



1 1 3 3
1 1 2 3
2 2 2 3
4 4 4 4



1 1 1 4
2 2 1 4
3 2 2 4
3 3 3 4



1 1 1 4
2 2 1 4
2 3 3 4
2 3 3 4
```



```
1 1 1 1
4 3 3 2
4 3 3 2
4 4 2 2



1 1 1 1
3 3 3 2
4 4 3 2
4 4 2 2



1 1 1 1
3 3 3 2
4 4 3 2
4 4 2 2



1 1 1 1
4 4 4 2
4 3 3 2
3 3 2 2



1 1 1 1
4 4 3 2
4 3 3 2
4 3 2 2



1 1 1 1
3 4 4 2
3 4 4 2
3 3 2 2
```



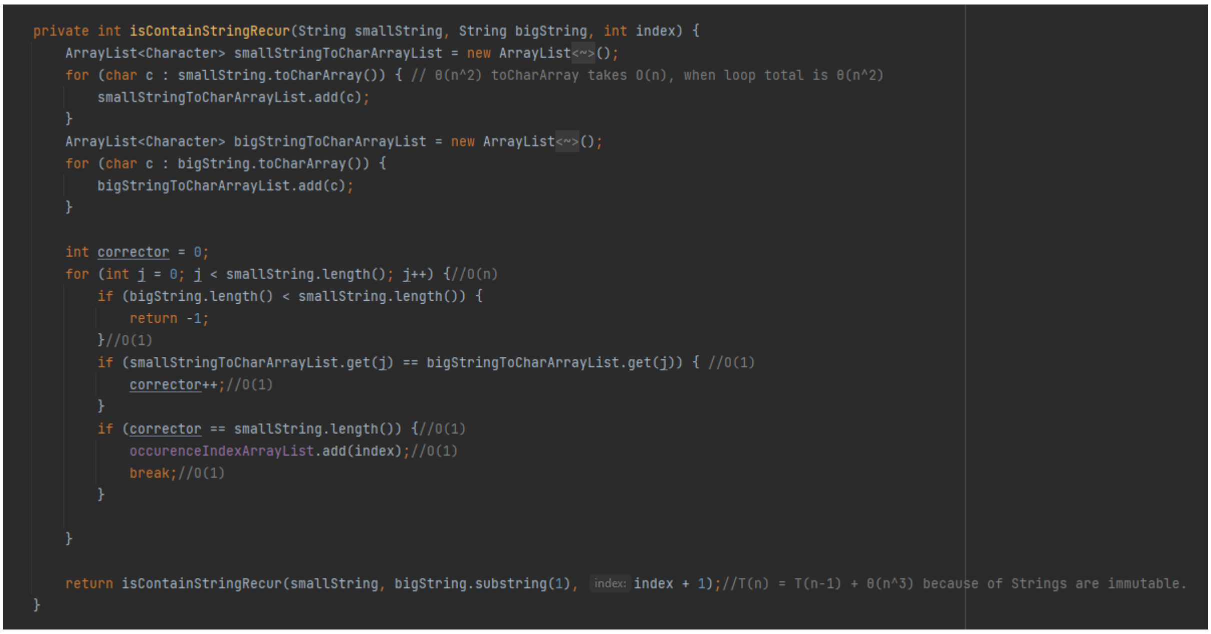
```
1 1 1 1
3 4 4 2
3 4 4 2
3 3 2 2



1 1 1 1
4 4 4 2
3 3 4 2
3 3 2 2



1 1 1 1
4 4 2 2
4 3 2 2
4 3 3 3



1 1 1 1
4 4 2 2
4 4 2 2
3 3 3 3



1 1 1 1
4 4 2 2
4 4 2 2
3 3 3 3



1 1 1 1
3 3 2 2
3 2 2 4
3 4 4 4
```



```
1 1 1 1
4 2 2 2
4 2 3 3
4 4 3 3



1 1 1 1
2 2 2 2
3 3 4 4
3 3 4 4



1 1 1 1
2 2 2 2
3 3 4 4
3 3 4 4



1 1 1 1
2 2 2 2
3 4 4 4
3 3 3 4



1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
```

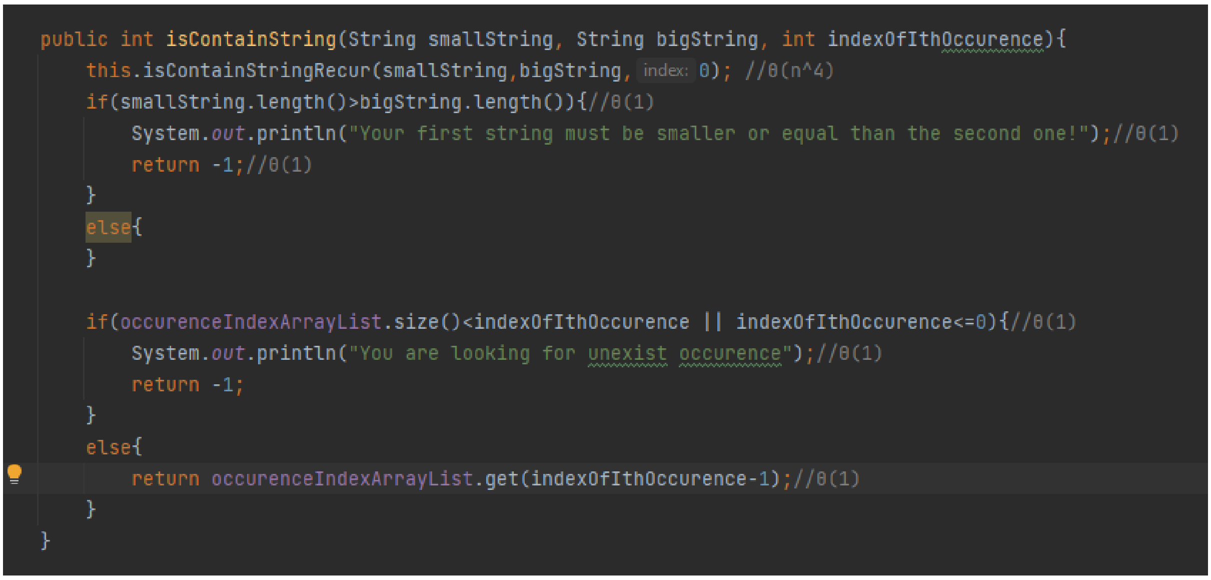
Pass

Time Complexities



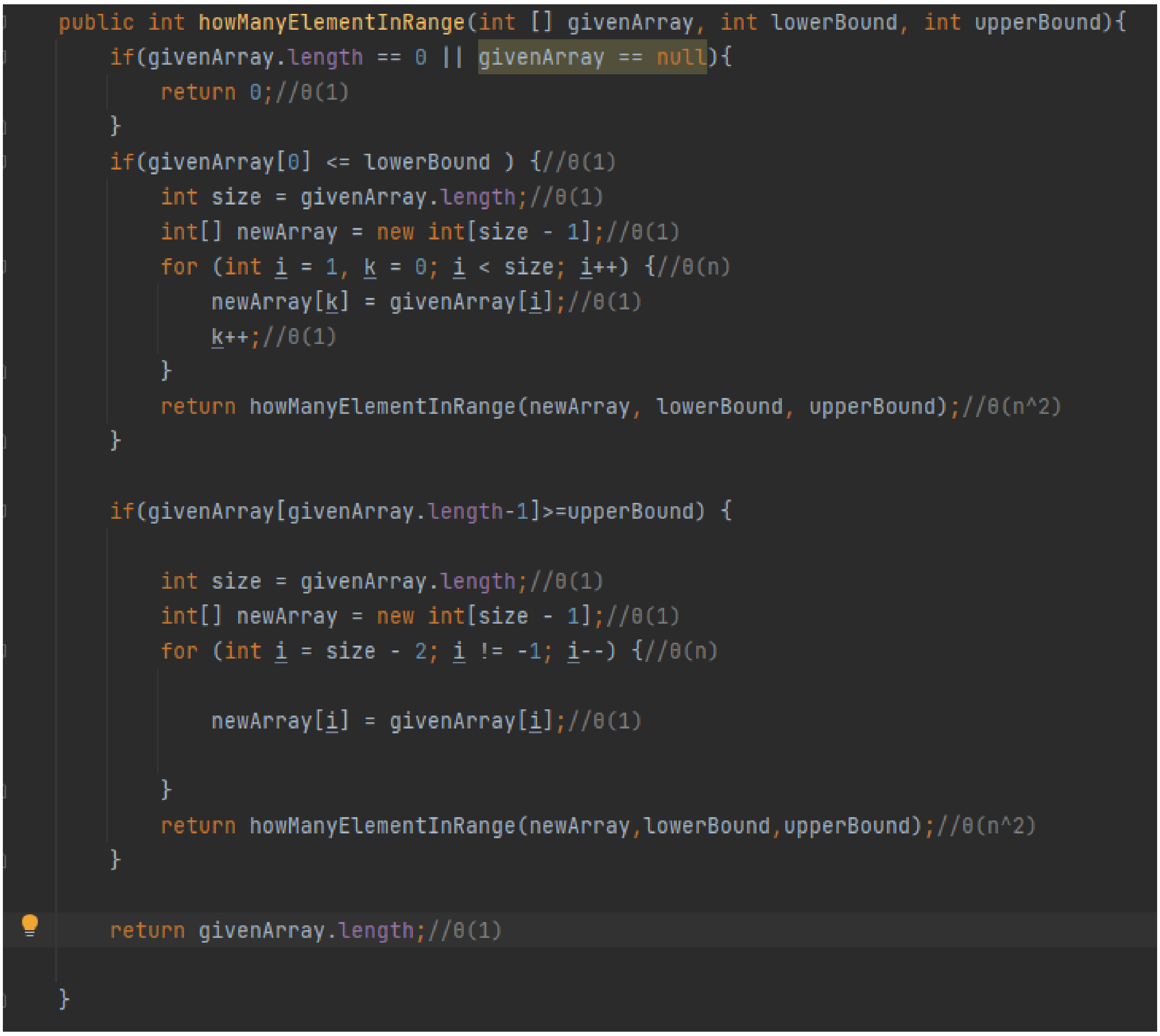
```java
private int isContainStringRecur(String smallString, String bigString, int index) {
    ArrayList<Character> smallStringToCharArrayList = new ArrayList<~>();
    for (char c : smallString.toCharArray()) { // θ(n^2) toCharArray takes O(n), when loop total is θ(n^2)
        smallStringToCharArrayList.add(c);
    }
    ArrayList<Character> bigStringToCharArrayList = new ArrayList<~>();
    for (char c : bigString.toCharArray()) {
        bigStringToCharArrayList.add(c);
    }

    int corrector = 0;
    for (int j = 0; j < smallString.length(); j++) {//O(n)
        if (bigString.length() < smallString.length()) {
            return -1;
        }//O(1)
        if (smallStringToCharArrayList.get(j) == bigStringToCharArrayList.get(j)) { //O(1)
            corrector++;//O(1)
        }
        if (corrector == smallString.length()) {//O(1)
            occurenceIndexArrayList.add(index);//O(1)
            break;//O(1)
        }

    }

    return isContainStringRecur(smallString, bigString.substring(1),  index: index + 1);//T(n) = T(n-1) + θ(n^3) because of Strings are immutable.
}
```

Θ(n^4)



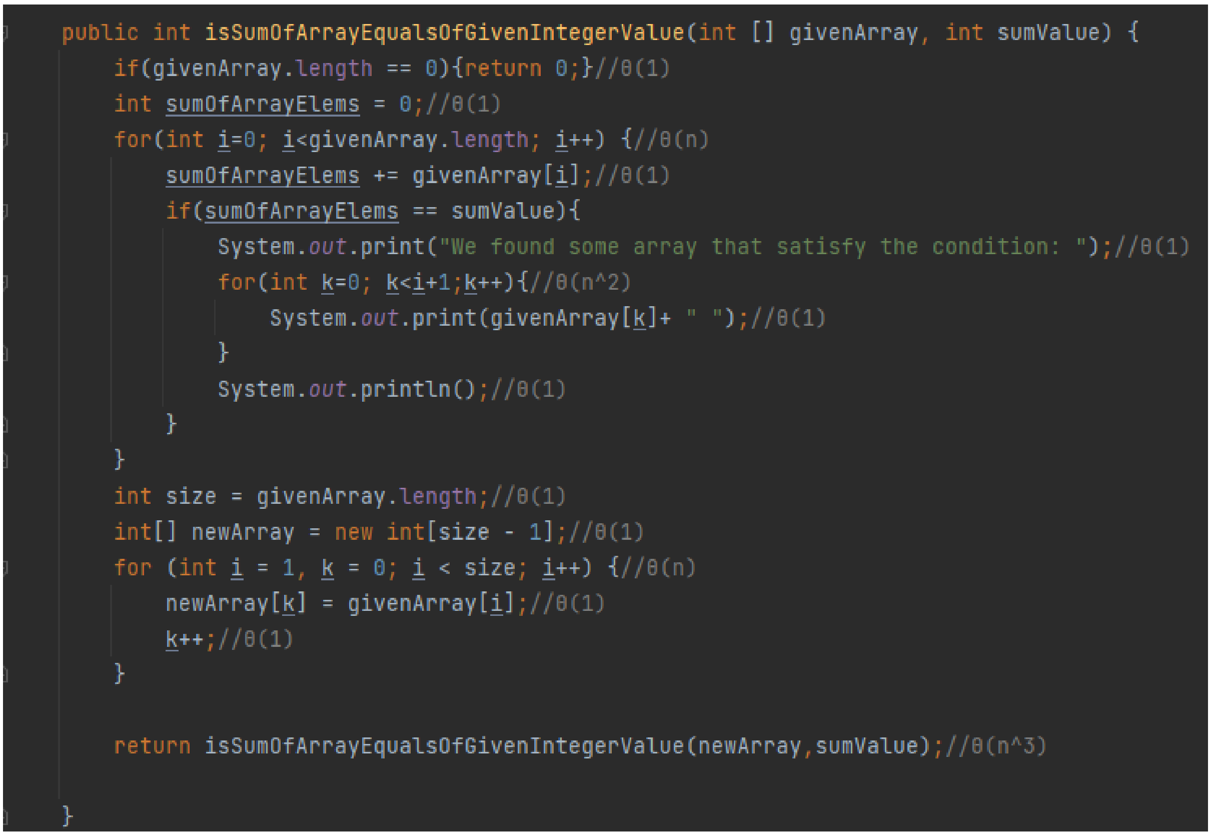
```java
public int isContainString(String smallString, String bigString, int indexOfIthOccurence){
    this.isContainStringRecur(smallString,bigString, index: 0); //θ(n^4)
    if(smallString.length()>bigString.length()){//θ(1)
        System.out.println("Your first string must be smaller or equal than the second one!");//θ(1)
        return -1;//θ(1)
    }
    else{
    }

    if(occurenceIndexArrayList.size()<indexOfIthOccurence || indexOfIthOccurence<=0){//θ(1)
        System.out.println("You are looking for unexist occurence");//θ(1)
        return -1;
    }
    else{
        return occurenceIndexArrayList.get(indexOfIthOccurence-1);//θ(1)
    }
}
```

Θ(n^4)



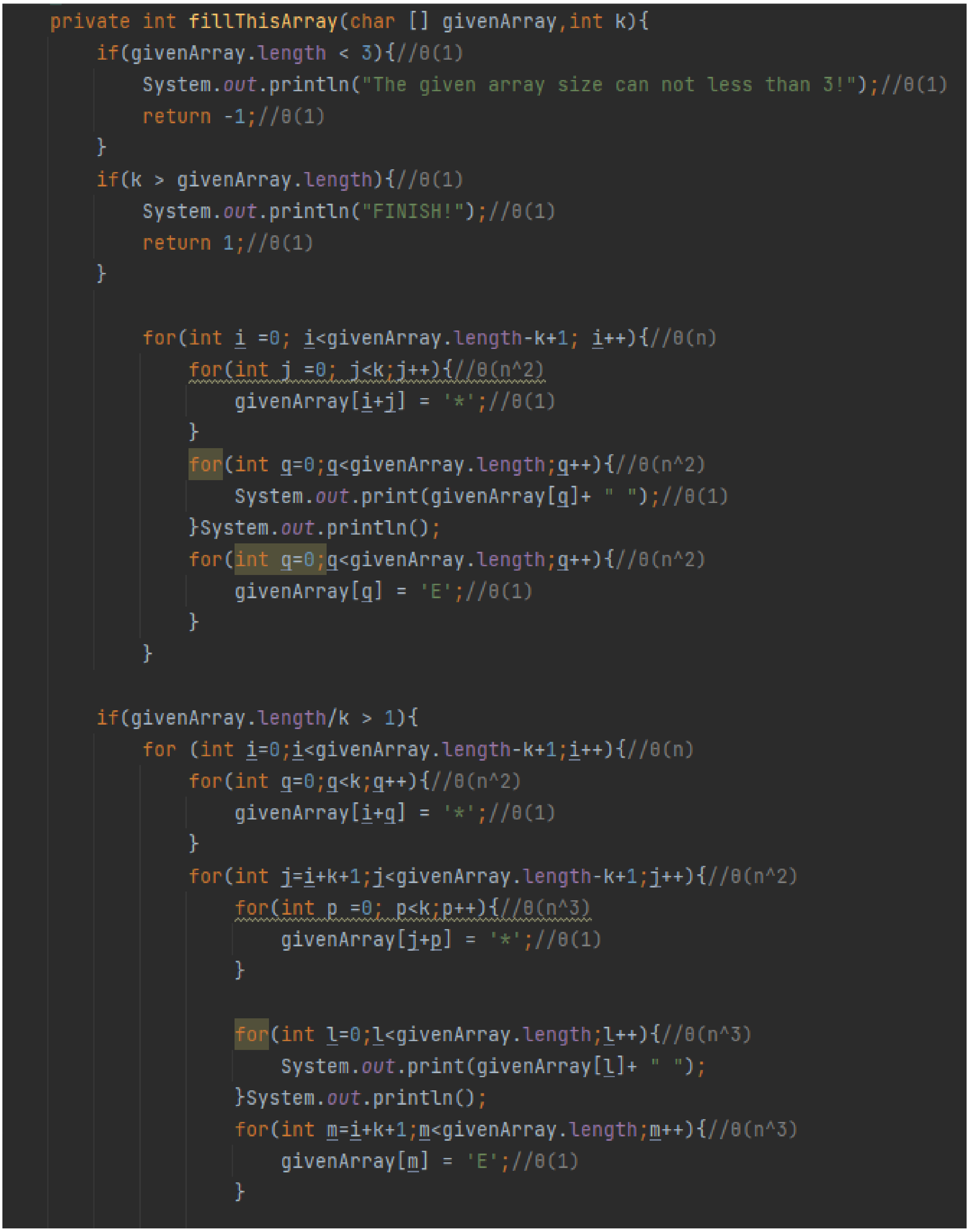
```java
public int howManyElementInRange(int [] givenArray, int lowerBound, int upperBound){
    if(givenArray.length == 0 || givenArray == null){
        return 0;//θ(1)
    }
    if(givenArray[0] <= lowerBound ) {//θ(1)
        int size = givenArray.length;//θ(1)
        int[] newArray = new int[size - 1];//θ(1)
        for (int i = 1, k = 0; i < size; i++) {//θ(n)
            newArray[k] = givenArray[i];//θ(1)
            k++;//θ(1)
        }
        return howManyElementInRange(newArray, lowerBound, upperBound);//θ(n^2)
    }

    if(givenArray[givenArray.length-1]>=upperBound) {

        int size = givenArray.length;//θ(1)
        int[] newArray = new int[size - 1];//θ(1)
        for (int i = size - 2; i != -1; i--) {//θ(n)

            newArray[i] = givenArray[i];//θ(1)

        }
        return howManyElementInRange(newArray,lowerBound,upperBound);//θ(n^2)
    }

    return givenArray.length;//θ(1)

}
```

Θ(n^2)

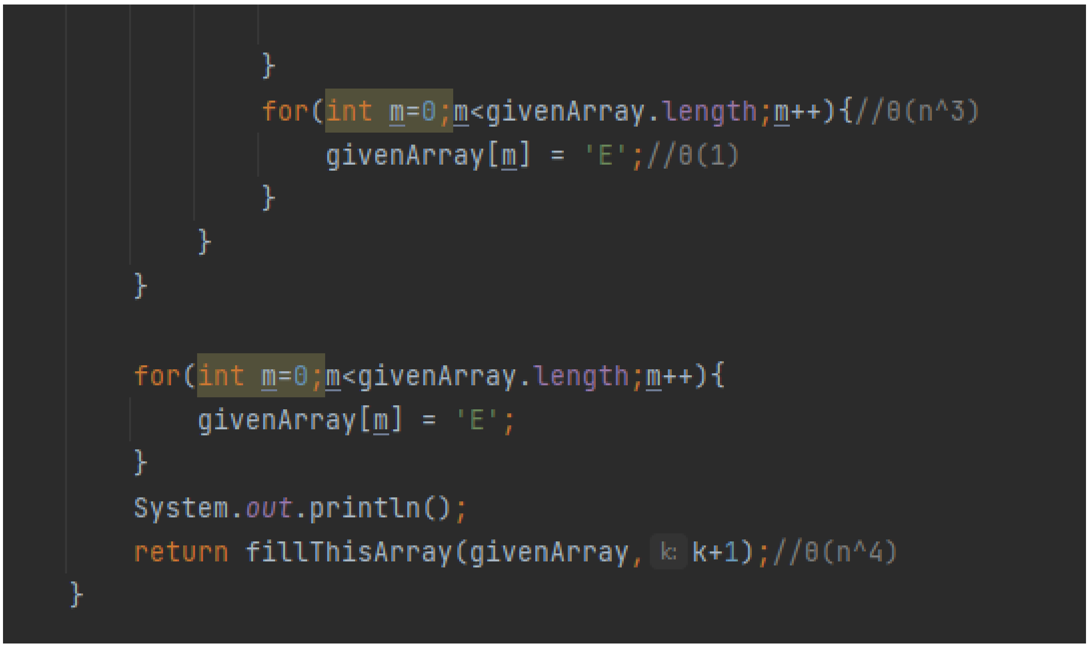


```
public int isSumOfArrayEqualsOfGivenIntegerValue(int [] givenArray, int sumValue) {
    if(givenArray.length == 0){return 0;}//θ(1)
    int sumOfArrayElems = 0;//θ(1)
    for(int i=0; i<givenArray.length; i++) {//θ(n)
        sumOfArrayElems += givenArray[i];//θ(1)
        if(sumOfArrayElems == sumValue){
            System.out.print("We found some array that satisfy the condition: ");//θ(1)
            for(int k=0; k<i+1;k++){//θ(n^2)
                System.out.print(givenArray[k]+ " ");//θ(1)
            }
            System.out.println();//θ(1)
        }
    }
    int size = givenArray.length;//θ(1)
    int[] newArray = new int[size - 1];//θ(1)
    for (int i = 1, k = 0; i < size; i++) {//θ(n)
        newArray[k] = givenArray[i];//θ(1)
        k++;//θ(1)
    }

    return isSumOfArrayEqualsOfGivenIntegerValue(newArray,sumValue);//θ(n^3)

}
```

Θ(n^3)

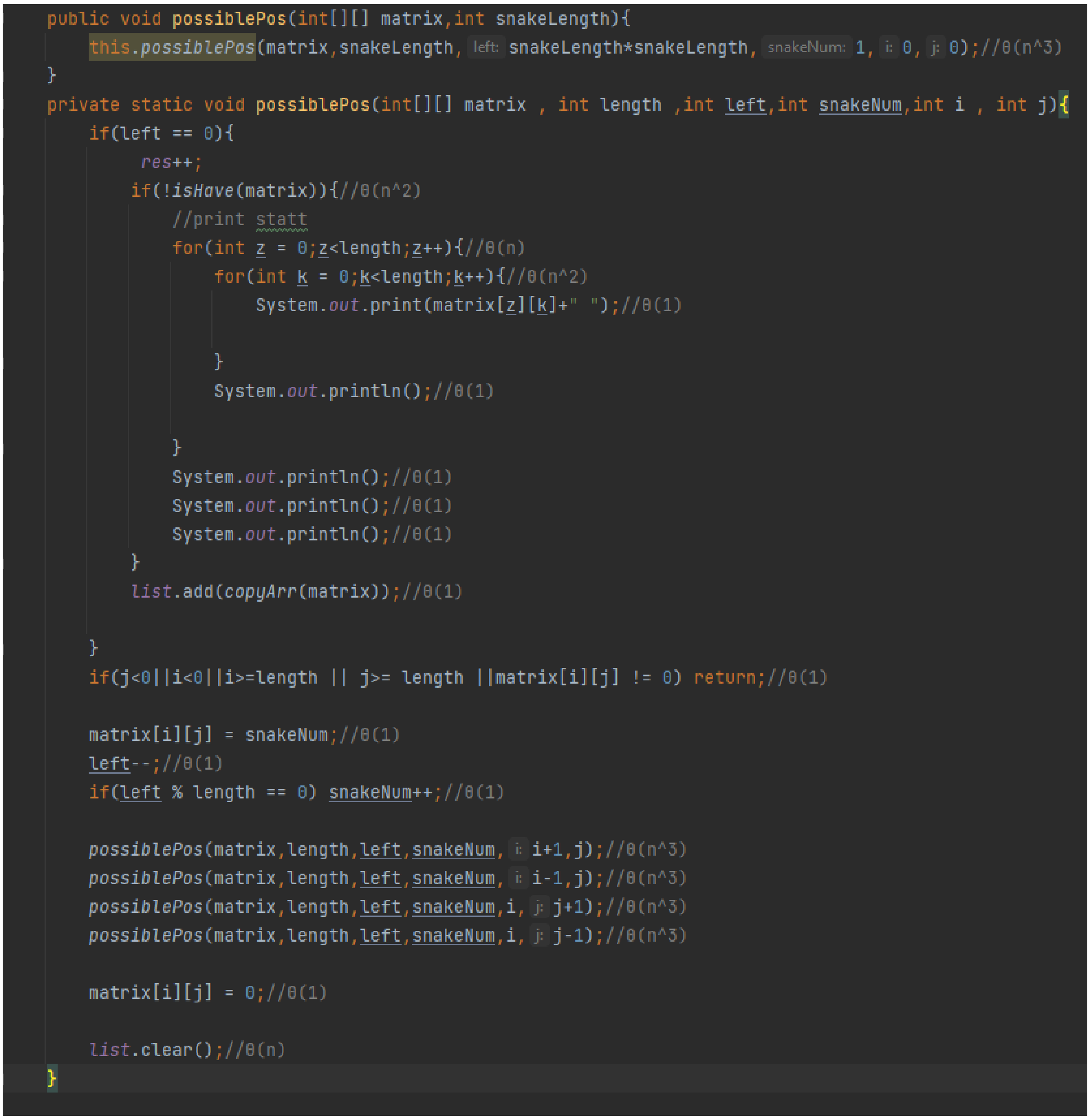


```
    private int fillThisArray(char [] givenArray,int k){
        if(givenArray.length < 3){//θ(1)
            System.out.println("The given array size can not less than 3!");//θ(1)
            return -1;//θ(1)
        }
        if(k > givenArray.length){//θ(1)
            System.out.println("FINISH!");//θ(1)
            return 1;//θ(1)
        }

            for(int i =0; i<givenArray.length-k+1; i++){//θ(n)
                for(int j =0; j<k;j++){//θ(n^2)
                    givenArray[i+j] = '*';//θ(1)
                }
                for(int q=0;q<givenArray.length;q++){//θ(n^2)
                    System.out.print(givenArray[q]+ " ");//θ(1)
                }System.out.println();
                for(int q=0;q<givenArray.length;q++){//θ(n^2)
                    givenArray[q] = 'E';//θ(1)
                }
            }

        if(givenArray.length/k > 1){
            for (int i=0;i<givenArray.length-k+1;i++){//θ(n)
                for(int q=0;q<k;q++){//θ(n^2)
                    givenArray[i+q] = '*';//θ(1)
                }
                for(int j=i+k+1;j<givenArray.length-k+1;j++){//θ(n^2)
                    for(int p =0; p<k;p++){//θ(n^3)
                        givenArray[j+p] = '*';//θ(1)
                    }

                    for(int l=0;l<givenArray.length;l++){//θ(n^3)
                        System.out.print(givenArray[l]+ " ");
                    }System.out.println();
                    for(int m=i+k+1;m<givenArray.length;m++){//θ(n^3)
                        givenArray[m] = 'E';//θ(1)
                    }
```



```
            }
            for(int m=0;m<givenArray.length;m++){//θ(n^3)
                givenArray[m] = 'E';//θ(1)
            }
        }
    }

    for(int m=0;m<givenArray.length;m++){
        givenArray[m] = 'E';
    }
    System.out.println();
    return fillThisArray(givenArray, k: k+1);//θ(n^4)
}
```

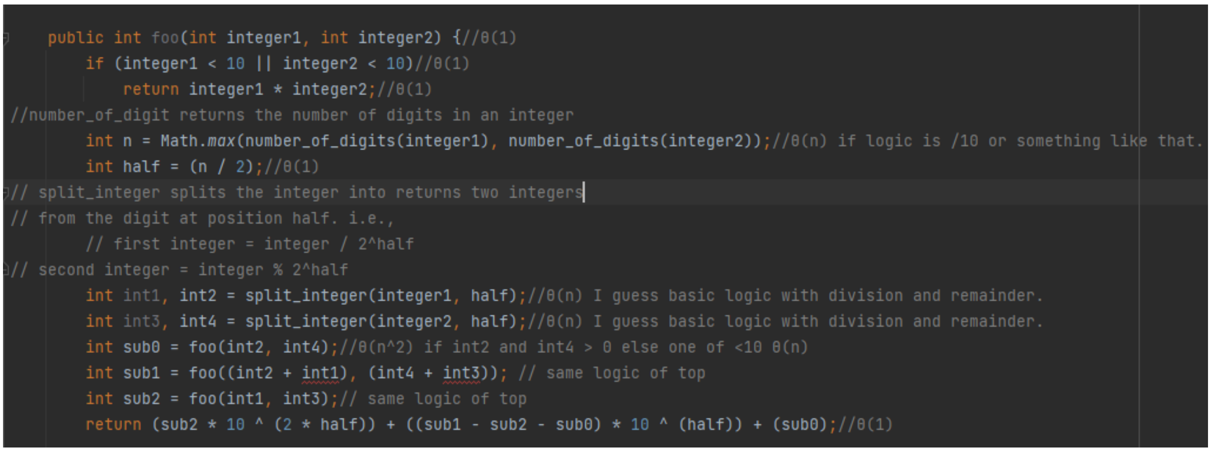
Θ(n^4)



```
public void possiblePos(int[][] matrix,int snakeLength){
    this.possiblePos(matrix,snakeLength, left: snakeLength*snakeLength, snakeNum: 1, i: 0, j: 0);//θ(n^3)
}
private static void possiblePos(int[][] matrix , int length ,int left,int snakeNum,int i , int j){
    if(left == 0){
         res++;
        if(!isHave(matrix)){//θ(n^2)
            //print statt
            for(int z = 0;z<length;z++){//θ(n)
                for(int k = 0;k<length;k++){//θ(n^2)
                    System.out.print(matrix[z][k]+" ");//θ(1)

                }
                System.out.println();//θ(1)

            }
            System.out.println();//θ(1)
            System.out.println();//θ(1)
            System.out.println();//θ(1)
        }
        list.add(copyArr(matrix));//θ(1)

    }
    if(j<0||i<0||i>=length || j>= length ||matrix[i][j] != 0) return;//θ(1)

    matrix[i][j] = snakeNum;//θ(1)
    left--;//θ(1)
    if(left % length == 0) snakeNum++;//θ(1)

    possiblePos(matrix,length,left,snakeNum, i: i+1,j);//θ(n^3)
    possiblePos(matrix,length,left,snakeNum, i: i-1,j);//θ(n^3)
    possiblePos(matrix,length,left,snakeNum,i, j: j+1);//θ(n^3)
    possiblePos(matrix,length,left,snakeNum,i, j: j-1);//θ(n^3)

    matrix[i][j] = 0;//θ(1)

    list.clear();//θ(n)
}
```

Θ(n^3)

Q4 - Time Complexity



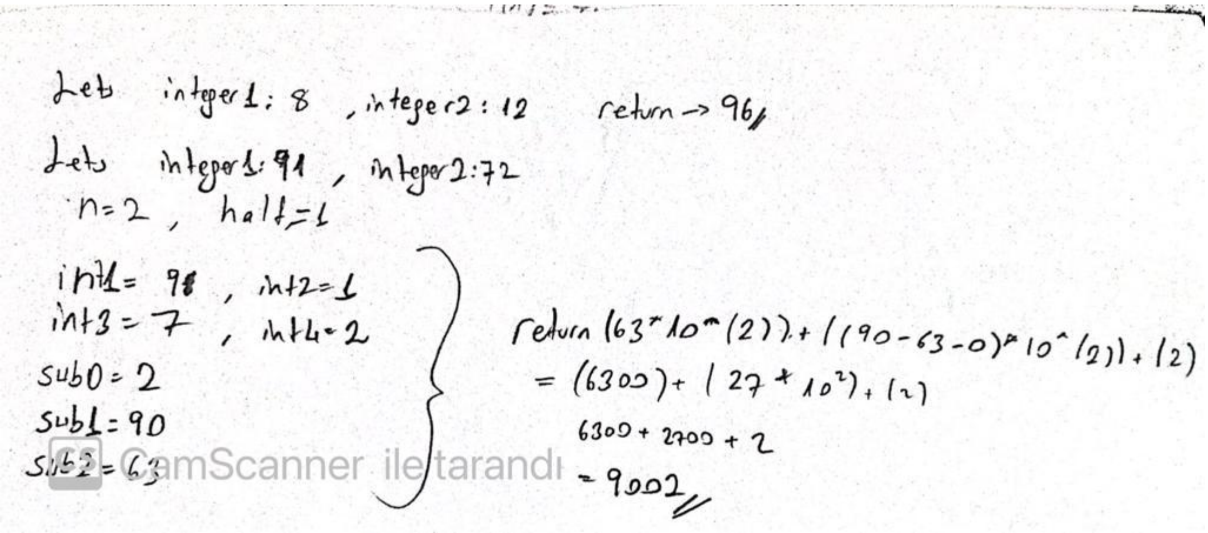
```
    public int foo(int integer1, int integer2) {//θ(1)
        if (integer1 < 10 || integer2 < 10)//θ(1)
            return integer1 * integer2;//θ(1)
//number_of_digit returns the number of digits in an integer
        int n = Math.max(number_of_digits(integer1), number_of_digits(integer2));//θ(n) if logic is /10 or something like that.
        int half = (n / 2);//θ(1)
// split_integer splits the integer into returns two integers
// from the digit at position half. i.e.,
        // first integer = integer / 2^half
// second integer = integer % 2^half
        int int1, int2 = split_integer(integer1, half);//θ(n) I guess basic logic with division and remainder.
        int int3, int4 = split_integer(integer2, half);//θ(n) I guess basic logic with division and remainder.
        int sub0 = foo(int2, int4);//θ(n^2) if int2 and int4 > 0 else one of <10 θ(n)
        int sub1 = foo((int2 + int1), (int4 + int3)); // same logic of top
        int sub2 = foo(int1, int3);// same logic of top
        return (sub2 * 10 ^ (2 * half)) + ((sub1 - sub2 - sub0) * 10 ^ (half)) + (sub0);//θ(1)
```

Best Case: Θ(1)

Worst Case: Θ(n^2)

Time Complexity: O(n^2)

FOR Q4-



Lets integer1: 8 , integer2: 12 return -> 96//

Lets integer1: 91 , integer2: 72

n=2 , half=1

int1 = 9 , int2 = 1

int3 = 7 , int4 = 2

sub0 = 2

sub1 = 90

sub2 = 63

return (63 * 10^(2)) + ((90 - 63 - 0) * 10^(2)) + (2)

= (6300) + (27 * 10^2) + (2)

6300 + 2700 + 2

= 9002//

Induction method proofs

For my HowManyElementInRange(int[] givenArray), int lowerBand, int upperBand)

$T(n) = T(n-1) + \Theta(n)$, when $c > 0$

our guess also is $T(n) \le kn^2$. for some $k > 0$ all n sufficiently large.

$$T(n) \le T(n-1) + cn$$
$$T(n) \le T(n-1) + cn$$
$$T(n) \le k(n-1)^2 + cn$$

and we require that $T(n) \le k(n-1)^2 + cn \le kn^2$

$$k(n-1)^2 + cn \le kn^2$$
$$k(n^2 - 2n+1) + cn \le kn^2$$
$$kn^2 - 2kn + k + cn \le kn^2$$
$$-2kn + k + cn \le 0$$
$$k(1-2n) \le -cn$$
$$k(2n-1) \ge cn$$
$$k \ge \frac{cn}{2n-1}$$

SO, since the largest value of $n/(2n-1)$ for $n \ge 1$ is 1 any $k \ge c$ will work. See this does work with $k=c$.

$$\begin{aligned} T(n) &\le T(n-1) + cn \\ &\le c(n-1)^2 + cn \\ &= cn^2 - 2cn + c + cn \\ &= cn^2 - cn + c \\ &= c(n^2 - n + 1) \le cn^2 \end{aligned}$$ → it proved!

On the other meaning,

$n \ge 1$ so $T(n) = O(n^2)$ ✓