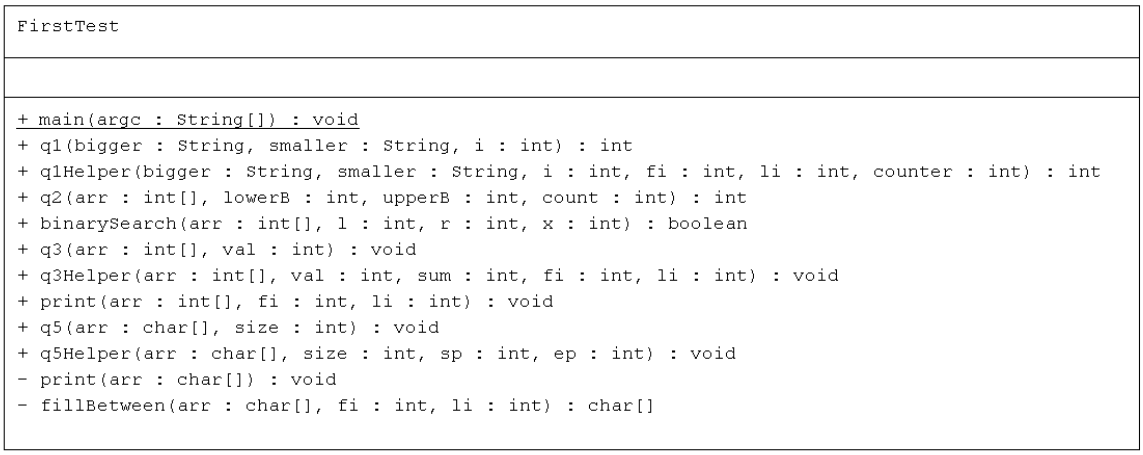
System Requirements:

- Proper jdk-jre
- Make command use

Class Diagram:

I Codded all in one class



Q1:

Problem solution approach:

The q1Helper function firstly checks that given index (i) is proper or not. After that checks last index (li) is out of bound or not. If there is no problem then compares the smaller string with sub string of bigger string between fi and li. If it's equal return fi index in case of that is the proper index that we want to take

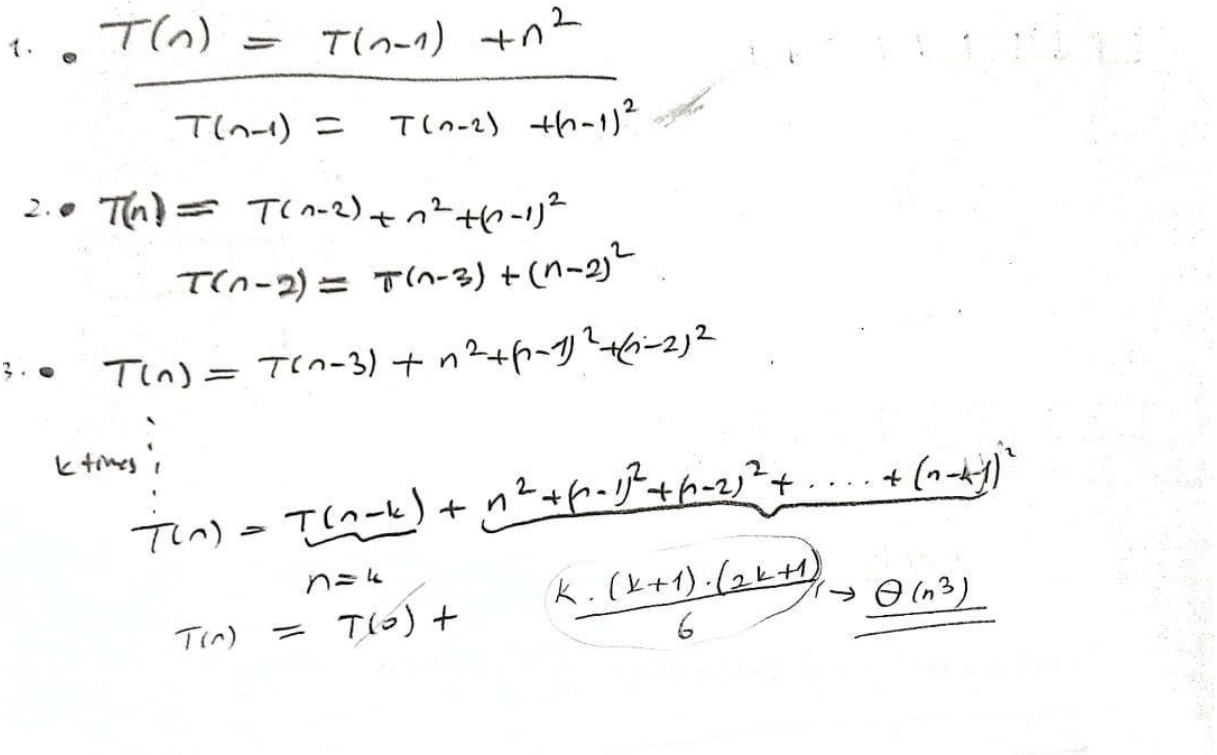
```
static int q1(String bigger, String smaller, int i) {

    return q1Helper(bigger,smaller,i,0,smaller.length(),0);
}

static int q1Helper(String bigger, String smaller, int i, int fi, int li, int counter ) {
    if(i<0) return -1;
    else if(li> bigger.length()) return -1;
    else if(bigger.substring(fi, li).equals(smaller)) {
        if(i==0) return fi;
        else if(i==counter) return fi;
        counter++;
    }

    return q1Helper(bigger,smaller,i,++fi,++li,counter);
}
```

Time Complexity:

```java
static int q1(String bigger, String smaller, int i) {

    return q1Helper(bigger,smaller,i,0,smaller.length(),0);
}

static int q1Helper(String bigger, String smaller, int i, int fi, int li, int counter ) {
    if(i<0) return -1;
    else if(li> bigger.length()) return -1;
    else if(bigger.substring(fi, li).equals(smaller)) {
        if(i==0) return fi;
        else if(i==counter) return fi;
        counter++;
    }

    return q1Helper(bigger,smaller,i,++fi,++li,counter);
}
```

→ $O(n^2)$

→ $T(n-1)$



Best: Theta(1)

Worst: Theta(n^3)

So **T(n) = O(n^3)**

Test cases and outputs:

```java
/*Q1 TEST*/
String q1 = "aabbcHacicmvHaci";
System.out.println("Q1 Answer: ");
System.out.println(obj.q1(q1,"Haci",0)); //output: 5
System.out.println(obj.q1(q1,"Haci",1)); //output: 12
System.out.println(obj.q1(q1,"Haci",2)); //output: -1
System.out.println(obj.q1(q1,"Hasan",0)); //output: -1
```

```
Q1 Answer:
5
12
-1
-1
```

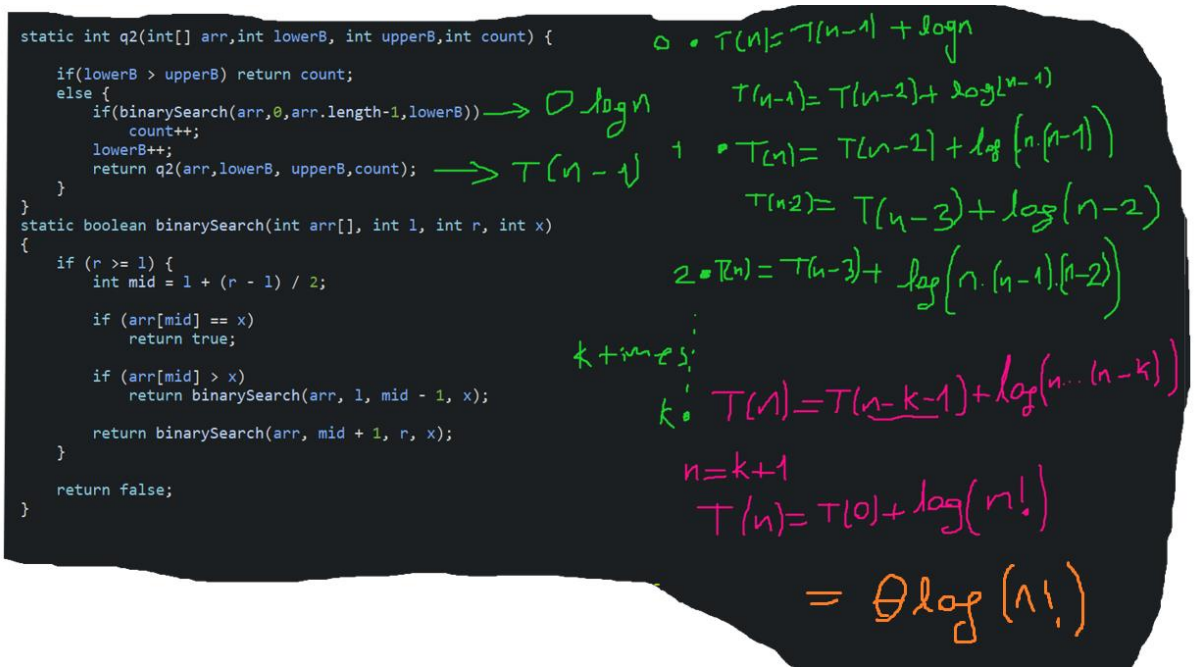
Q2:

Problem solution approach:

q2 method travels between lower Bound (lowerB) and upper Bound (upperB) and uses classic binary search algorithm to check that is in the array.

```java
static int q2(int[] arr,int lowerB, int upperB,int count) {

    if(lowerB > upperB) return count;
    else {
        if(binarySearch(arr,0,arr.length-1,lowerB))
            count++;
        lowerB++;
        return q2(arr,lowerB, upperB,count);
    }
}
```

```java
static boolean binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;

        if (arr[mid] == x)
            return true;

        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);

        return binarySearch(arr, mid + 1, r, x);
    }

    return false;
}
```



Best: Theta(1)

Worst: Theta(log(n!))

So **T(n) = O(log(n!))**

Induction analysis:

Q3/ $T(n) = T(n-1) + \log n \stackrel{?}{=} O(\log(n!))$

$T(0) = 0$

$n=1 \rightarrow T(1) = T(0) + \log 1 = 0$

$n=2 \rightarrow T(2) = T(1) + \log 2 = \log 2$

$n=3 \rightarrow T(3) = T(2) + \log(3) = \log 6$

$\vdots$

Induction hypothesis: Assume that for some arbitrary value of n that $T(n) = \log(n!)$ is true

$T(n+1) = T(n) + \log(n+1)$

$\log(n!)$

$T(n+1) \Rightarrow \log((n+1) \cdot n!) = \log((n+1)!)$

Test cases and outputs:

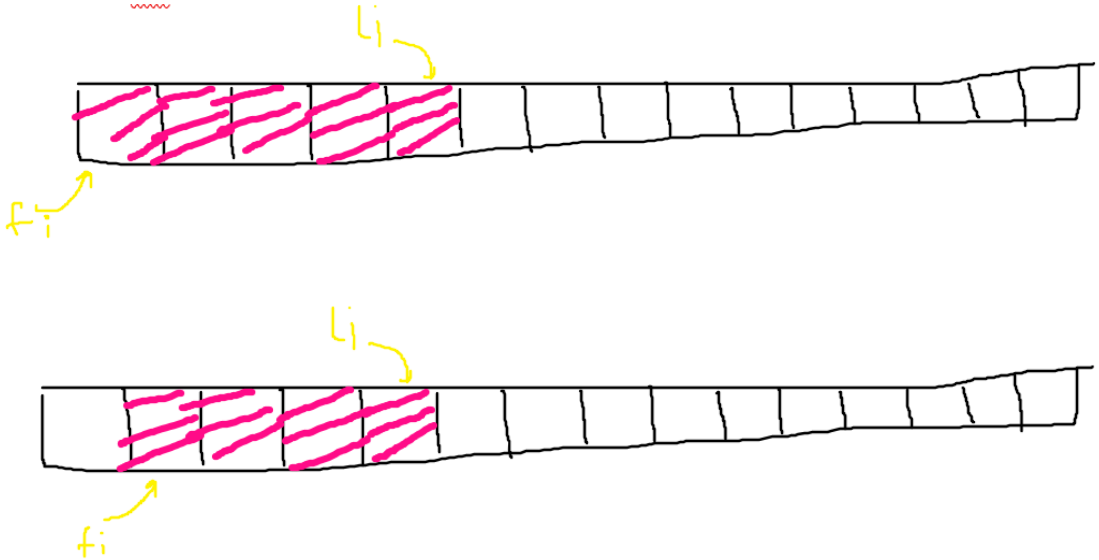
```
/*Q2 TEST*/
int[] arr = {10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,180};
int count = obj.q2(arr,2,62,0); //result: 6
System.out.println("Q2 Answer1: "+ count);

int count2 = obj.q2(arr,12,179,0);
System.out.println("Q2 Answer2: "+ count2); //result: 16
System.out.println("\n----------------------");
```

```
----------------------
Q2 Answer1: 6
Q2 Answer2: 16

----------------------
```

Q3:

Problem solution approach:

the q3Helper function travels the array with fi (first index) and li (last index). In every iteration the arr[fi] will be added to the sum. If sum is equal to the val prints between li and fi and then continue.

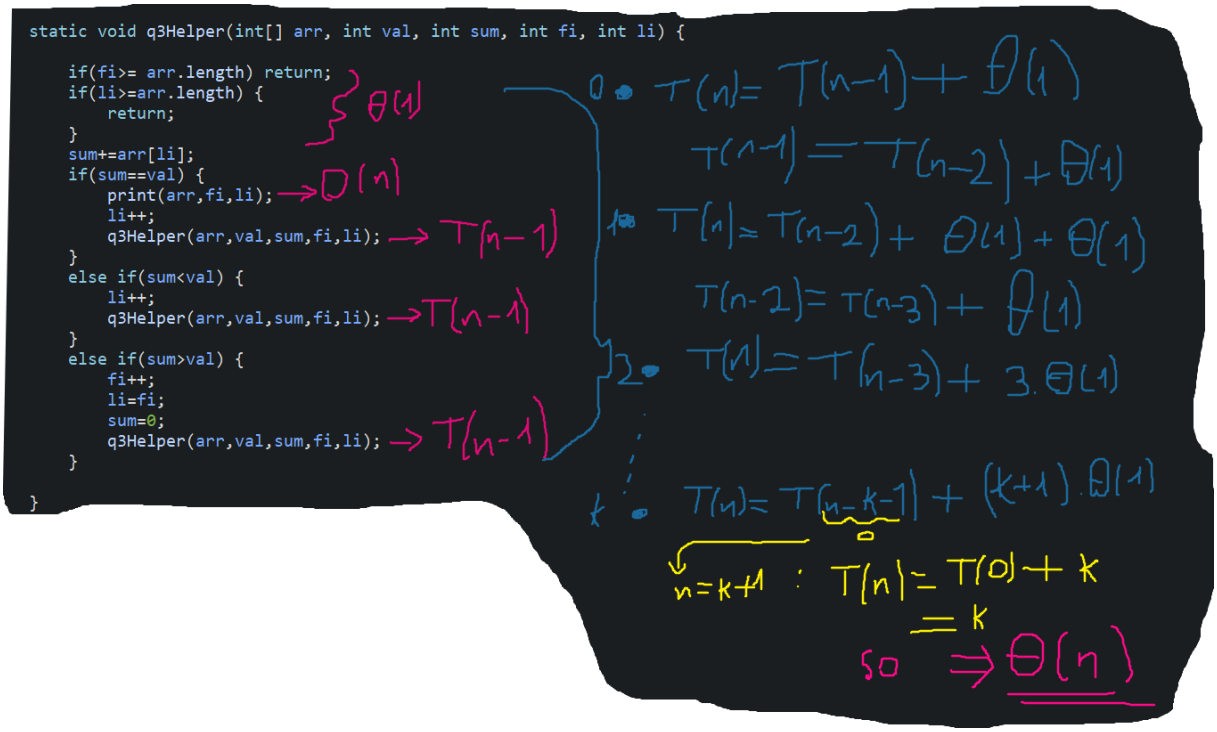


In every iteration checks like above

```
static void q3(int[] arr, int val) {
    q3Helper(arr,val,0,0,0);
}
```

```java
static void q3Helper(int[] arr, int val, int sum, int fi, int li) {

    if(fi>= arr.length) return;
    if(li>=arr.length) {
        return;
    }
    sum+=arr[li];
    if(sum==val) {
        print(arr,fi,li);
        li++;
        q3Helper(arr,val,sum,fi,li);
    }
    else if(sum<val) {
        li++;
        q3Helper(arr,val,sum,fi,li);
    }
    else if(sum>val) {
        fi++;
        li=fi;
        sum=0;
        q3Helper(arr,val,sum,fi,li);
    }

}
```

```java
 static void print(int[] arr, int fi, int li) {
     if(fi>= arr.length || li>= arr.length) {

         System.out.println();
         return;
     }
     if(fi>li) {
         System.out.println();
         return;
     }
     System.out.print(arr[fi]+" ");
     print(arr,++fi,li);
 }
```

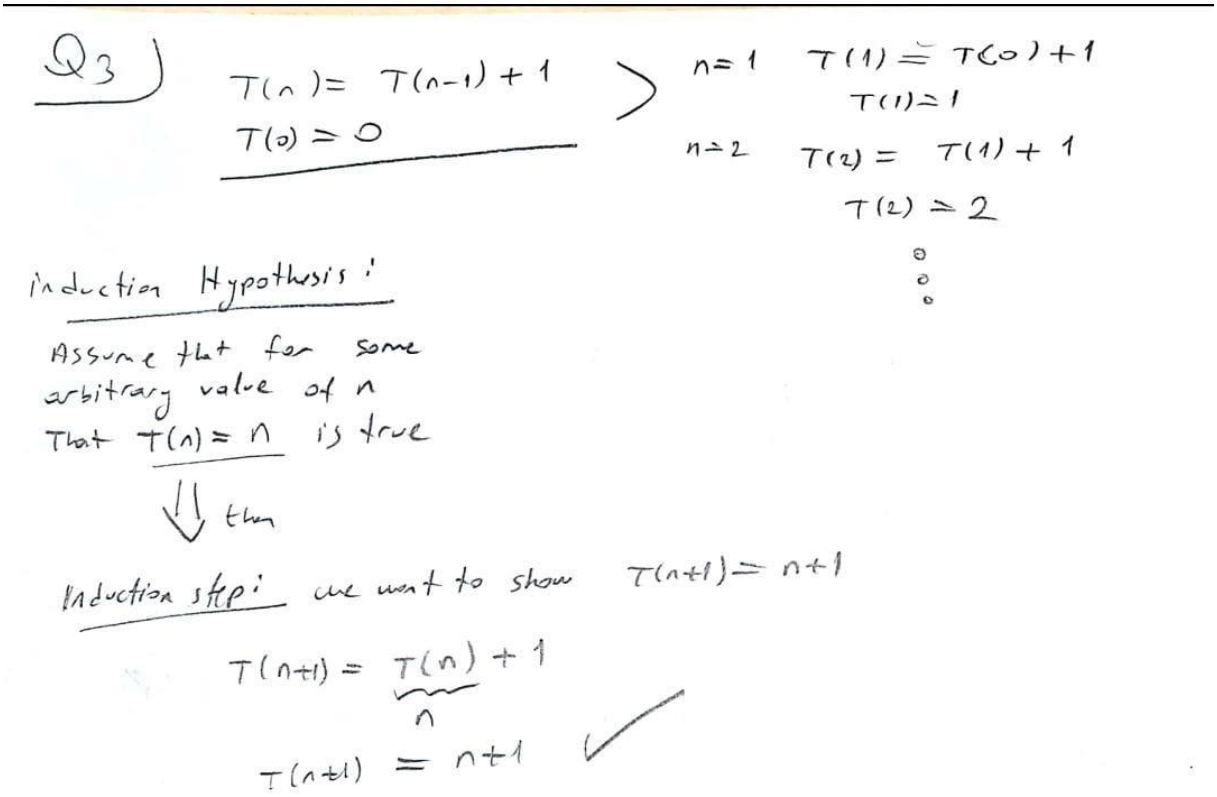


Best: Theta(1)

Worst: Theta(n)

So **T(n) = O(n)**

Induction analysis:



Test cases and outputs:

```
/*Q3 TEST*/
int[] arr3 = {5,3,8,1,2,11,0,7,5,6,8,4,12,13,15,17,20,2,0,11,10,1,0,1};
System.out.println("Q3 Answer1: (sum: 11)");
obj.q3(arr3,11);
System.out.println("-----");
System.out.println("Q3 Answer2: (sum: 12)");
obj.q3(arr3,12);
```

```
----------------------
Q3 Answer1: (sum: 11)
3 8
8 1 2
11
11 0
5 6
0 11
11
10 1
10 1 0
-----
Q3 Answer2: (sum: 12)
3 8 1
0 7 5
7 5
8 4
12
10 1 0 1
```

Q4:

- This recursive function returns multiplication of two given integr

Example work :

integer_1 = 10 , integer_2 = 20

n=2
half=1

int1 = 1 int2 = 0
int3 = 2 int4 = 0

sub0 = foo(0,0) = 0
sub1 = foo(1,2) = 2
sub2 = foo(1,2) = 2

return $2 \cdot 10^2 + \cancel{(2-2-0)} \cdot 10^{\overset{0}{2}} + 0$
$= 200$ ✓

Base case :
integer_1 or integer_2 is less than 10

- $T(n) = 3 \cdot T(n/2) + n$

  $T(n/2) = 3T(n/4) + n/2$

- $T(n) = 3^2 \cdot T(n/4) + n + n/2$

  ⋮ ⋮

$T(n) = 3^k + T(n/2^k) + \sum_{i=1}^{k} \frac{n}{i2}$

$n = 2^k$
$k = \log n$

$= 3^{\log n} = \Theta(n^{\log 3})$

Q5)

Problem solution approach:

The function travels in the array and keeps the first and last index of sub arrays. The first and last indexes are increased one by one. And function checks if there is enough space for the next same size sub array. If there is enough space to put one more then first and last indexes adjusted to the next proper place

The q5Helper function is

```java
/**
 * Just uses the q5Helper(....) method
 * @param arr char array
 * @param size start size
 */
public void q5(char[] arr, int size){
    Arrays.fill(arr, '.');
    q5Helper(arr,size,0,size-1);
}
```

```java
/**
 * The given array must will be traveled
 * @param arr char array
 * @param size start size.
 * @param sp start point
 * @param ep end point
 */
public void q5Helper(char[] arr, int size, int sp, int ep){

    if(ep==arr.length) {
        Arrays.fill(arr, '.'); //clear the array
        size++;
        System.out.println();

        if(size<=arr.length){

            q5Helper(arr,size,0,size-1);
        }
    }
    else{

        float rest = (float)(arr.length-ep-1)/size;
        /*arr[sp] = 'o';
        arr[ep] = 'o';*/
        arr = fillBetween(arr, sp, ep);
        if(rest>1){
            int nsp = ep+2;
            int nep = ep+size+1;
            //arr[nsp] = '#';
            //arr[nep] = '#';
            arr = fillBetween(arr, nsp, nep);
        }
        print(arr);
        Arrays.fill(arr, '.');
        sp++;
        ep++;
        q5Helper(arr,size,sp,ep);
    }
```

Just helper functions:

```java
/**
 * Just a print function
 * @param arr array to be print
 */
private void print(char[] arr){
    for(int i =0; i< arr.length; ++i)
        System.out.print(arr[i]+" ");
    System.out.println();

}
/**
 * Just fills between fi and li with 'o'
 * @param arr char array to be partially filled
 * @param fi first index
 * @param li last index
 * @return new partially filled array
 */
private char[] fillBetween(char[] arr, int fi,int li){
    if(li>= arr.length || fi>=arr.length){
        System.out.print("error");
        return null;
    }
    for(int i=fi; i<=li; ++i){
        arr[i] = 'o';
    }
    return arr;
}
```

Outputs and results:

```java
//Q5 TEST
System.out.println("\n----------------------");
System.out.println("Q5.1");
char[] arr2 = new char[7];
obj.q5(arr2,3);

System.out.println("\nQ5.2");
char[] arr3 = new char[9];
obj.q5(arr3,3);
```

```
Q5.1
o o o . o o o
. o o o . . .
. . o o o . .
. . . o o o .
. . . . o o o

o o o o . . .
. o o o o . .
. . o o o o .
. . . o o o o

o o o o o . .
. o o o o o .
. . o o o o o

o o o o o o .
. o o o o o o

o o o o o o o
```


Q5.2
o o o . o o o . .
. o o o . o o o .
. . o o o . o o o
. . . o o o . . .
. . . . o o o . .
. . . . . o o o .
. . . . . o o o
o o o o . o o o o
. o o o o . . . .
. . o o o o . . .
. . . o o o o . .
. . . . o o o o .
. . . . . o o o o
o o o o o . . . .
. o o o o o . . .
. . o o o o o . .
. . . o o o o o .
. . . . o o o o o
o o o o o o . . .
. o o o o o o . .
. . o o o o o o .
. . . o o o o o o
o o o o o o o . .
. o o o o o o o .
. . o o o o o o o
o o o o o o o o .
. o o o o o o o o
o o o o o o o o o