GIT Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 4 Report

Tuba Toprak
161044116

1. SYSTEM REQUIREMENTS

User requirements and Functional Requirements

In this assignment, the following operations were performed when the user run the system.

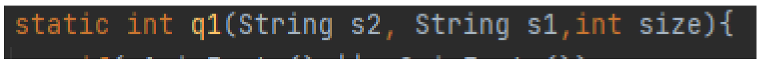
1- finding a small string in a big string

2- Finding the numbers between the given range in a sorted array

3- Finding subsets adjacent to the given sum in an unordered array**.**

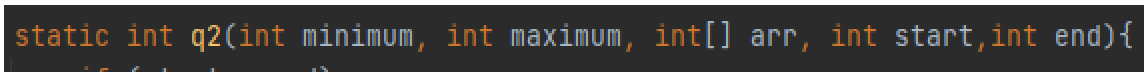
5- calculates all the possible configurations to fill array with colored-blocks with length at least 3

In Question 1: In this question, the recursion function takes a large string and a small string. The operation here returns 0 if the big string starts with a small string. If it doesn't, the function repeats itself by deleting the first letter of the big string. If the searched small string does not exist, the function returns -1. In this process, it is done with string methods (startsWith() , substring()) without using any additional libraries.



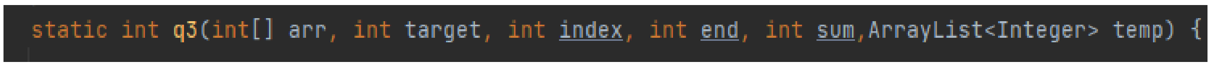
```
static int q1(String s2, String s1,int size){
```

In Question 2: In this question, the function takes two numbers as minimum and maximum to determine ranges, sorted array and two numbers as start and end to traverse the sorted array. A midpoint is expected within the function and the array is searched with binary search logic and compared with the given range. Detailed information is in problem solution approach. No additional library was used.



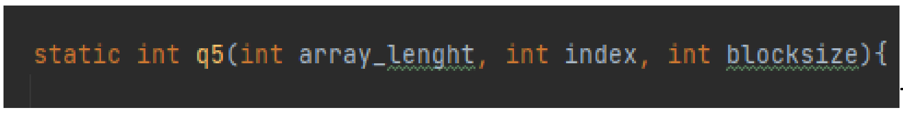
```
static int q2(int minimum, int maximum, int[] arr, int start,int end){
```

In Question 3: The function takes as arguments an unordered array, the searched sum, the index of the array, the end number to navigate the array, and an arraylist to store the numbers. The first call is made as arraylist is empty, end and index 0.



```
static int q3(int[] arr, int target, int index, int end, int sum,ArrayList<Integer> temp) {
```

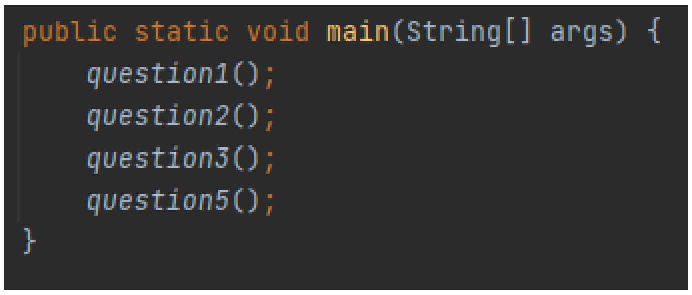
In Question 5: As arguments to this function, it takes length of array , the size of the block, and the number of indexes to be able to recursive.



```
static int q5(int array_lenght, int index, int blocksize){
```

This function makes a temporary arraylist and adds 0 to the size of the array. Then the base case was written and the function was sent to the recursive function for both rows and columns.

Each program is divided into sub-functions. Users can try whatever they want or run them all at once.



```
public static void main(String[] args) {
    question1();
    question2();
    question3();
    question5();
}
```

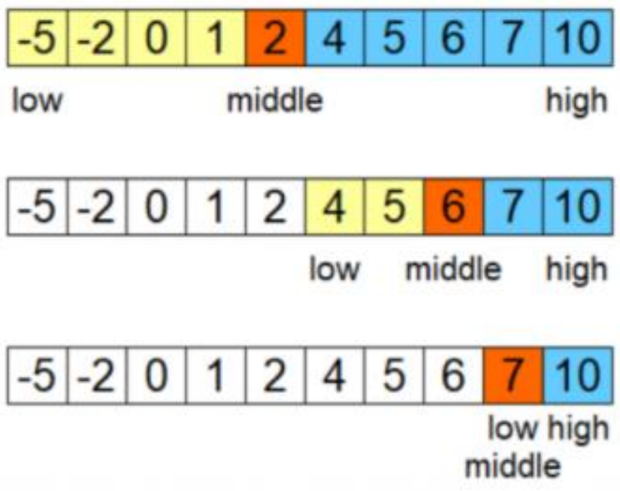
Software requirements

This system is designed in IntelliJ idea. Version is java 15.0.2. The software runs on any operating system and IDE.

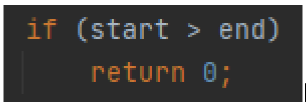
2. PROBLEM SOLUTION APPROACH

In Question 2:

The logic in the picture you see below has been followed. A solution has been reached by reducing the arrays so that there are no numbers left. Here, the index shown in orange in each reduction operation has been checked.



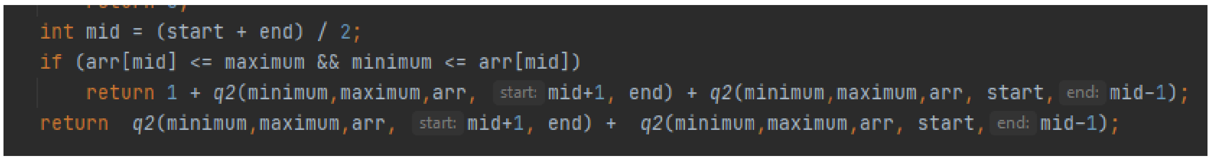
After specifying the arguments in the second question, the base case was written. The goal is to drive a starting point to the end and finish after specifying the arguments the base case was written.



```
if (start > end)
    return 0;
```

Here it is checked whether the array is empty.

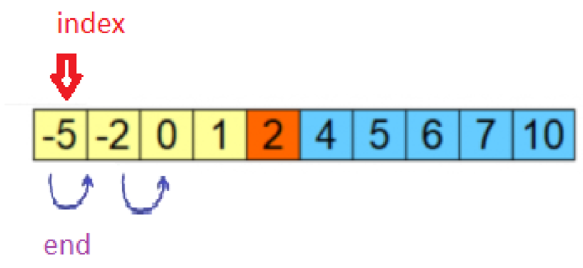
It then sets a midpoint, splits the array in half, and returns the two arrays back to their functions.



```
int mid = (start + end) / 2;
if (arr[mid] <= maximum && minimum <= arr[mid])
    return 1 + q2(minimum,maximum,arr, start: mid+1, end) + q2(minimum,maximum,arr, start, end: mid-1);
return  q2(minimum,maximum,arr, start: mid+1, end) +  q2(minimum,maximum,arr, start, end: mid-1);
```

What it checks here is that if the midpoint number is in the range, it is sent as +1.

In Question 3: Here it is understood that n*n crawls will be made for each index.
The purpose is to compare the index with all indexes and to make it do this to all indexes.
Here the index repeats itself unchanged until the end number reaches the end of the array.
If the end of the array is reached, the index is increased by one and the end number is returned to the beginning and the same process is applied again.



3. COMLEXITY ANALYSIS AND QUESTION 4

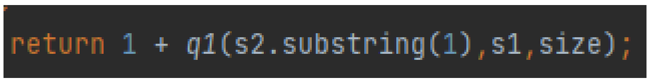
In question 1 : in best cases Θ(1) 3. İf Θ(1)
T(n) = T(n-1) + Θ(1) = Θ(n) best case: Θ(1) worst case: Θ(n) Complexity Time :O(n)

In Question 2: In base case : Θ(1) 2. İf: Θ(1) else: T(n) = Θ(log n) best case: Θ(1) worst case: Θ(log n) complexity Time: O(log n)

In Question 3**:** In base Cases: Θ(1) 3. İf : Θ(1) else: Θ(n) Best case: Θ(n) worst case: Θ(n) Complexity Time: Θ(n)

In Question 5**:** In Base cases: Θ(1) for loop: Θ(n) print function : O(n) Recursive Function: T(n)= T(n-1) + Θ(n) = Θ(n^2) other recursive function: Θ(n^2)
Best case: Θ(1) Worst Case : Θ(n^2) Complexity Time : O(n^2)

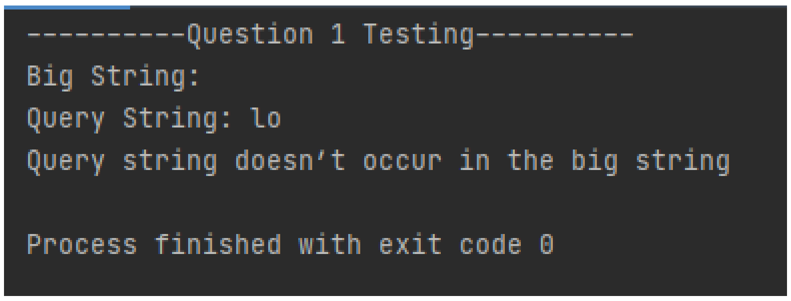
Proof Question 1 with By Induction: Best case = n; n == str1.isEmpty() so n = 0;
It has been proven that the theorem is true for the base case.
It has been shown that the theorem is accepted as true for n. Therefore, n+1 is true.



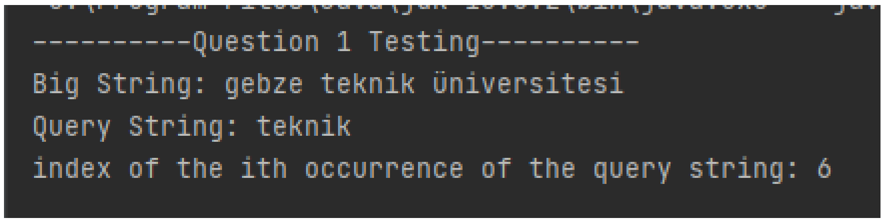
```
return 1 + q1(s2.substring(1),s1,size);
```

this is k +1 . Here, k decreases and goes to base case. We have proven the base case, that is, this operation adds +1 forward and finds the result.

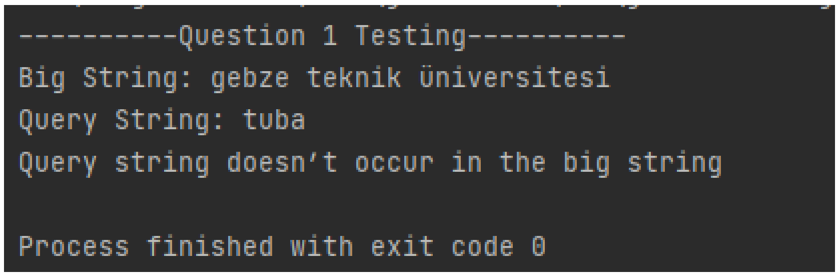
4. RUNNING AND RESULTS - TEST CASES



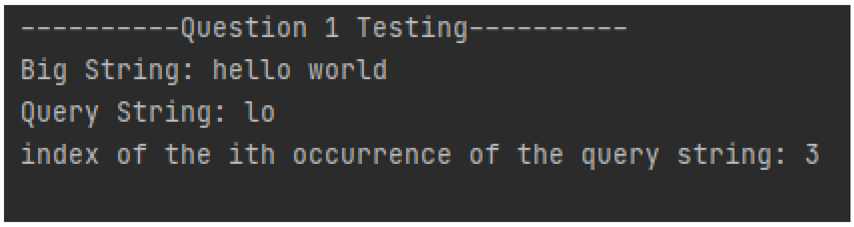
```
----------Question 1 Testing----------
Big String:
Query String: lo
Query string doesn’t occur in the big string

Process finished with exit code 0
```



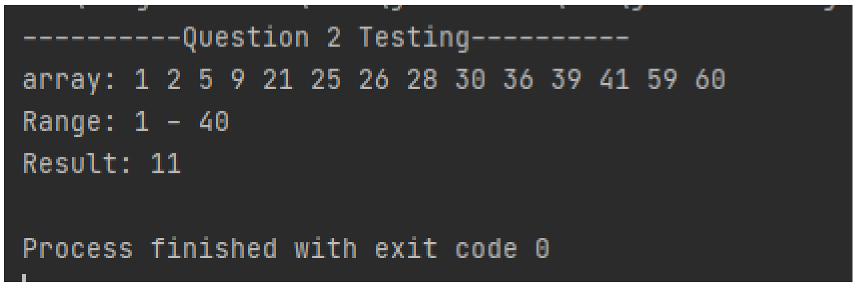
```
----------Question 1 Testing----------
Big String: gebze teknik üniversitesi
Query String: teknik
index of the ith occurrence of the query string: 6
```



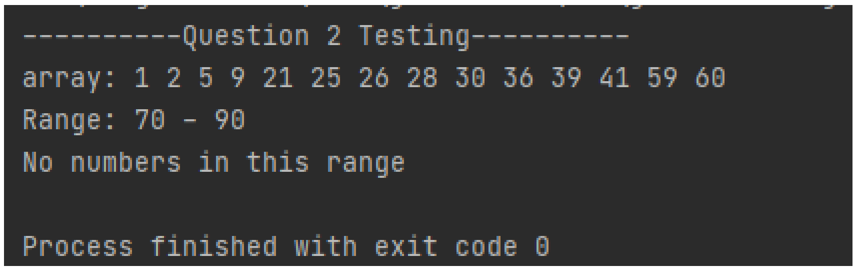
```
----------Question 1 Testing----------
Big String: gebze teknik üniversitesi
Query String: tuba
Query string doesn’t occur in the big string

Process finished with exit code 0
```



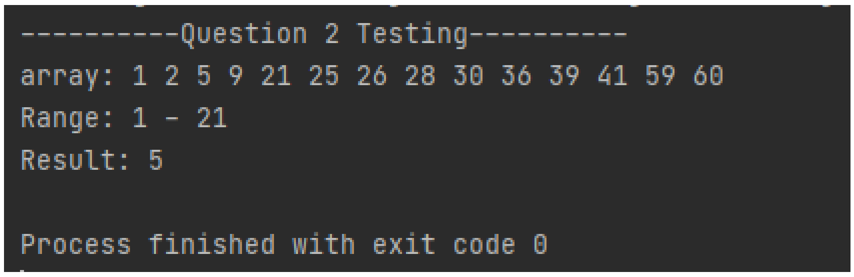
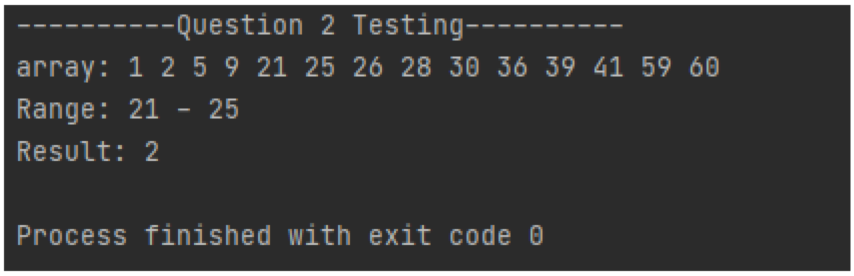
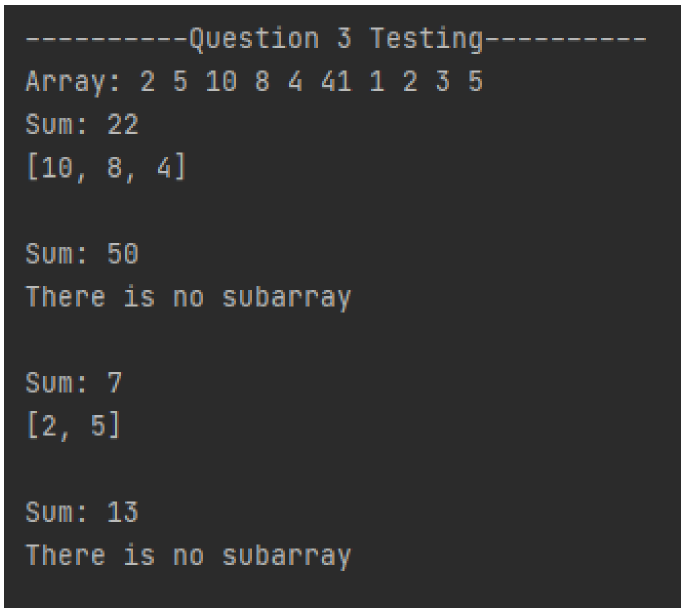
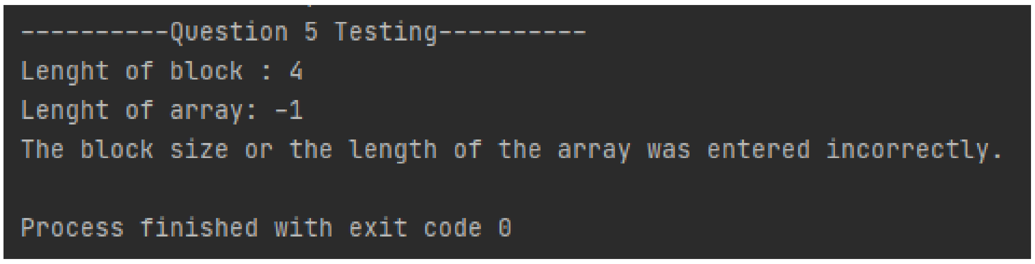
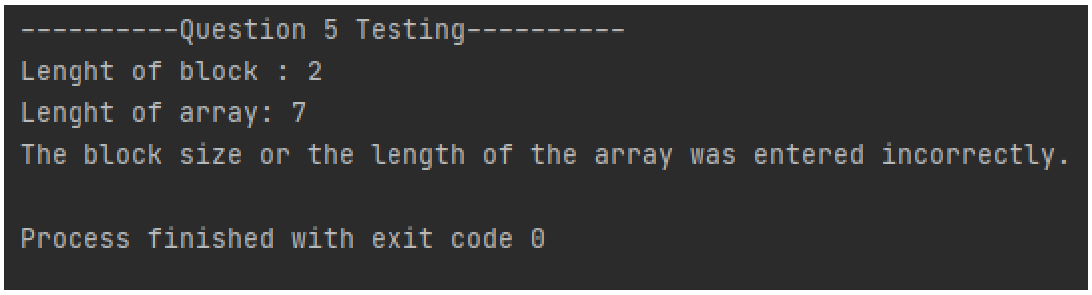
```
----------Question 1 Testing----------
Big String: hello world
Query String: lo
index of the ith occurrence of the query string: 3
```

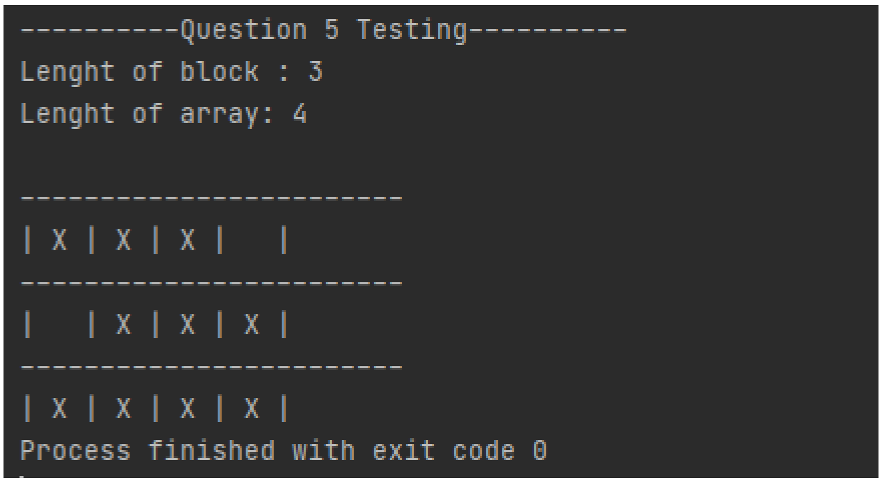
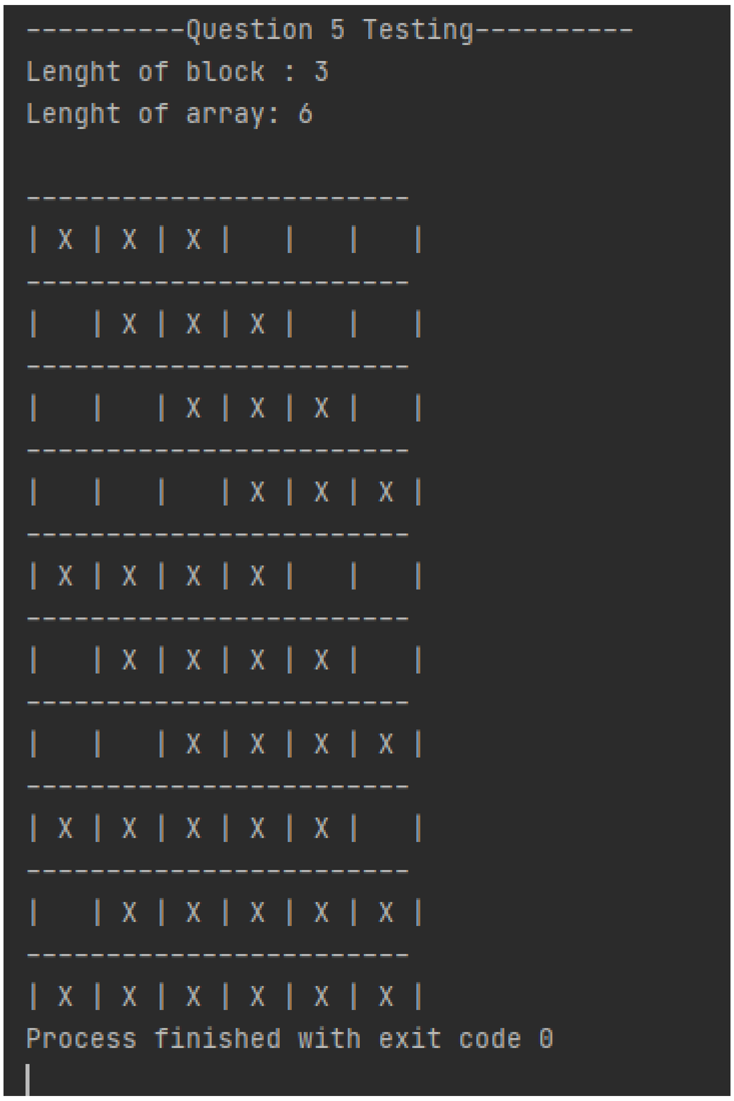


```
----------Question 2 Testing----------
array: 1 2 5 9 21 25 26 28 30 36 39 41 59 60
Range: 1 - 40
Result: 11

Process finished with exit code 0
```



```
----------Question 2 Testing----------
array: 1 2 5 9 21 25 26 28 30 36 39 41 59 60
Range: 70 - 90
No numbers in this range

Process finished with exit code 0
```


```
----------Question 2 Testing----------
array: 1 2 5 9 21 25 26 28 30 36 39 41 59 60
Range: 1 - 21
Result: 5

Process finished with exit code 0
```

```
----------Question 2 Testing----------
array: 1 2 5 9 21 25 26 28 30 36 39 41 59 60
Range: 21 - 25
Result: 2

Process finished with exit code 0
```

```
----------Question 3 Testing----------
Array: 2 5 10 8 4 41 1 2 3 5
Sum: 22
[10, 8, 4]

Sum: 50
There is no subarray

Sum: 7
[2, 5]

Sum: 13
There is no subarray
```

```
----------Question 5 Testing----------
Lenght of block : 4
Lenght of array: -1
The block size or the length of the array was entered incorrectly.

Process finished with exit code 0
```

```
----------Question 5 Testing----------
Lenght of block : 2
Lenght of array: 7
The block size or the length of the array was entered incorrectly.

Process finished with exit code 0
```


```
----------Question 5 Testing----------
Lenght of block : 3
Lenght of array: 4

------------------------
| X | X | X |   |
------------------------
|   | X | X | X |
------------------------
| X | X | X | X |
Process finished with exit code 0
```

```
----------Question 5 Testing----------
Lenght of block : 3
Lenght of array: 6

------------------------
| X | X | X |   |   |   |
------------------------
|   | X | X | X |   |   |
------------------------
|   |   | X | X | X |   |
------------------------
|   |   |   | X | X | X |
------------------------
| X | X | X | X |   |   |
------------------------
|   | X | X | X | X |   |
------------------------
|   |   | X | X | X | X |
------------------------
| X | X | X | X | X |   |
------------------------
|   | X | X | X | X | X |
------------------------
| X | X | X | X | X | X |
Process finished with exit code 0
```