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

Ahmet Kadir Aksu
200104004114

1. SYSTEM REQUIREMENTS

The system implements four different algorithms (which written recursively)

1- (Q1) static int findString(String str, String queryStr, int i)
Returns index of i th occurence of given string in a big string
And it is case sensitive (for example: if you are looking for "xy" it does not count "XY")
There need 2 strings for the method: query string and big string.

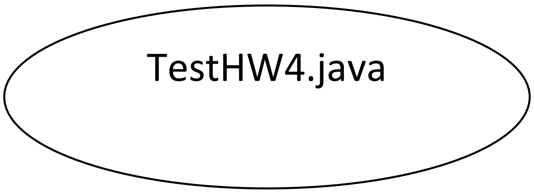
2-(Q2) static int numberOfItemsBetween(int arr[], int first, int second)
Returns the number of items between two number.
There need one sorted integer array and two integers for the method.

3-(Q3) static void findSubArrays(int[] arr, int sum)
Prints out contiguous subarray/s in the array that the sum of its/their items is equal to a given integer value.
There need an integer array and an integer for the method.

4-(Q5) static void colorBlocks(char[] blocks)
Prints all the possible configurations to fill the array with colored-blocks with length at least 3.
'C' represents colored blocks, while '0' represents non-colored blocks.
There needs a char array for the method.
The algorithm has missing parts.
So, it does not print the cases which there are more than one empty block between colored blocks
(i.e., It does not include CCC000CCC0 for the array with length 10)

2. USE CASE AND CLASS DIAGRAMS

There is no need any class diagram since there is only driver class for the program.



3. PROBLEM SOLUTION APPROACH

METHOD 1

For the first method, base case is either the big string does not contain the query string, or the *index* is 0 (Returns -1).

Next step is, it divides the string from the index of first occurrence of query string in the big string and it passes the new string into the method with decreasing index value by 1.

Meantime indexes will be added and finally it will return the answer.

The method will work T(n) = O(n^2) since string methods are already working with O(n).

METHOD 2

For the second method, the method returns -1 if the first number is bigger than the second (etc. if the first = 6, second = 4, it returns -1)

Then the method finds the index of both number by using binary search

If any of them (*first* or *second* variables) do not exist in the array, it assigns appropriate index, for the *first* it goes upper index, while it goes lower index for the *second.*

(What I mean is, let say array is: "2, 3, 5, 8, 10, 15" if *first* is 4, and *second* is 12, first index will be found as 2 and second index will be found as 3.)

Then, it returns (*lastIndex – firstIndex + 1*) which gives the answer.

The result includes the boundaries. (i.e. array is: 2 3 5 8, if first = 3 and second = 8, then the answer will be **3** (3,5,8))

Time complexity of the method will be O(log n) since we use binary search here.

METHOD 3

For the third method, it returns if the index (which I started from zero and then increase) equals the length of the array.

It keeps an integer *result* which keeps sum of the contiguous subarrays.

Also, it keeps a string to hold contiguous subarrays.

If the *result* is equals to the given *sum*, it prints the string and look for the new subarrays.

If the *result* is higher than the *sum*, resets the result and look for the new subarrays.

If the *result* is lower than the *sum*, it keeps adding the next element of the array.

It works O(n^3) since string summation is O(n).

METHOD 5

For the method 5,

The base case is, if the *coloredLength* (which is the length of the colored blocks, started from 3) is bigger than length of the array.

'C' represents colored blocks, while '0' represents non-colored blocks.

First it prints all the possibilities colored blocks with the length of 3.

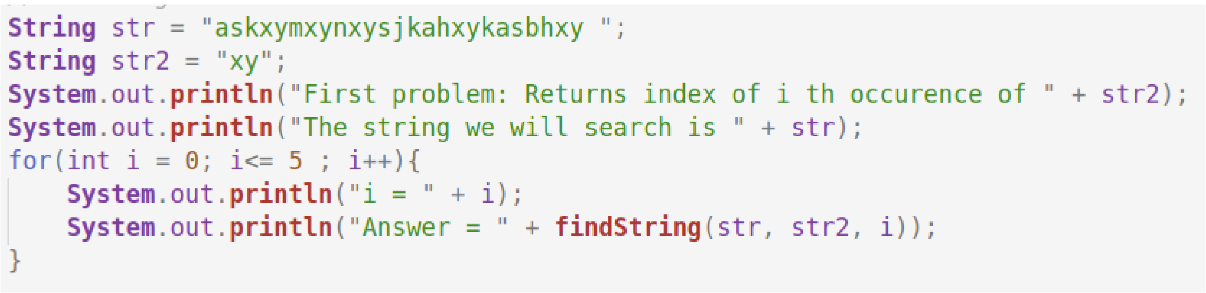
Then it increases the length

Algorithm has missing parts, it does not print the cases which there are more than one empty block between colored blocks.

4. TEST CASES

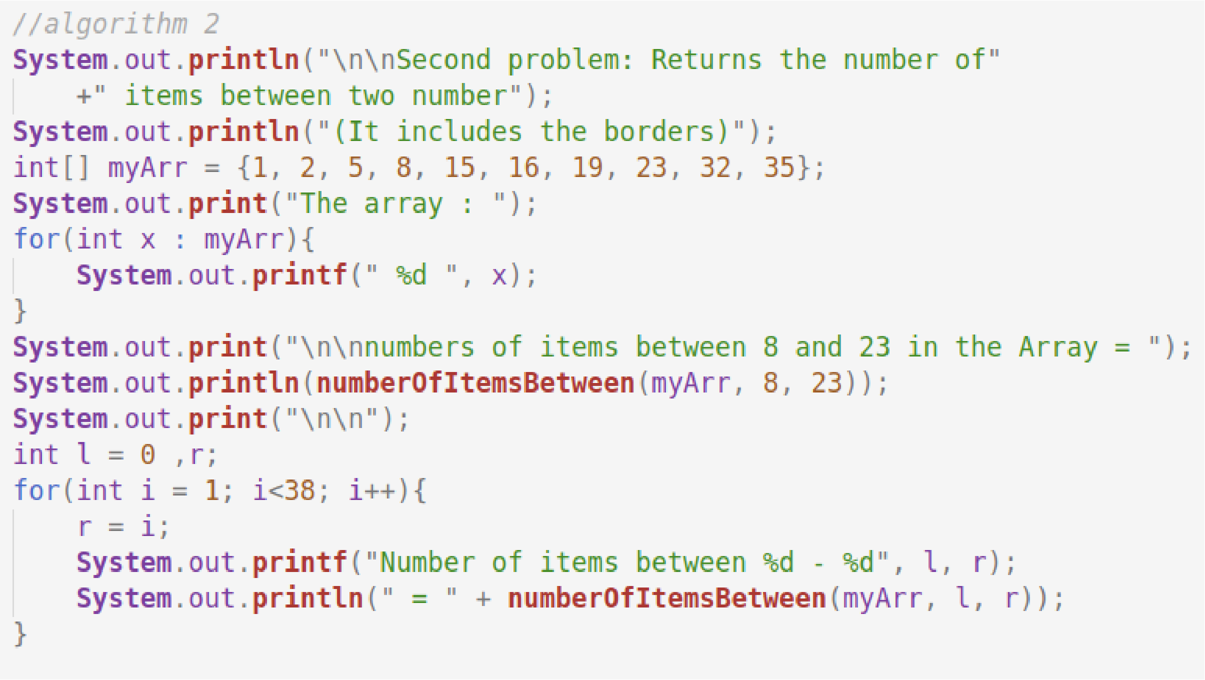
Test Cases

1st algorithm



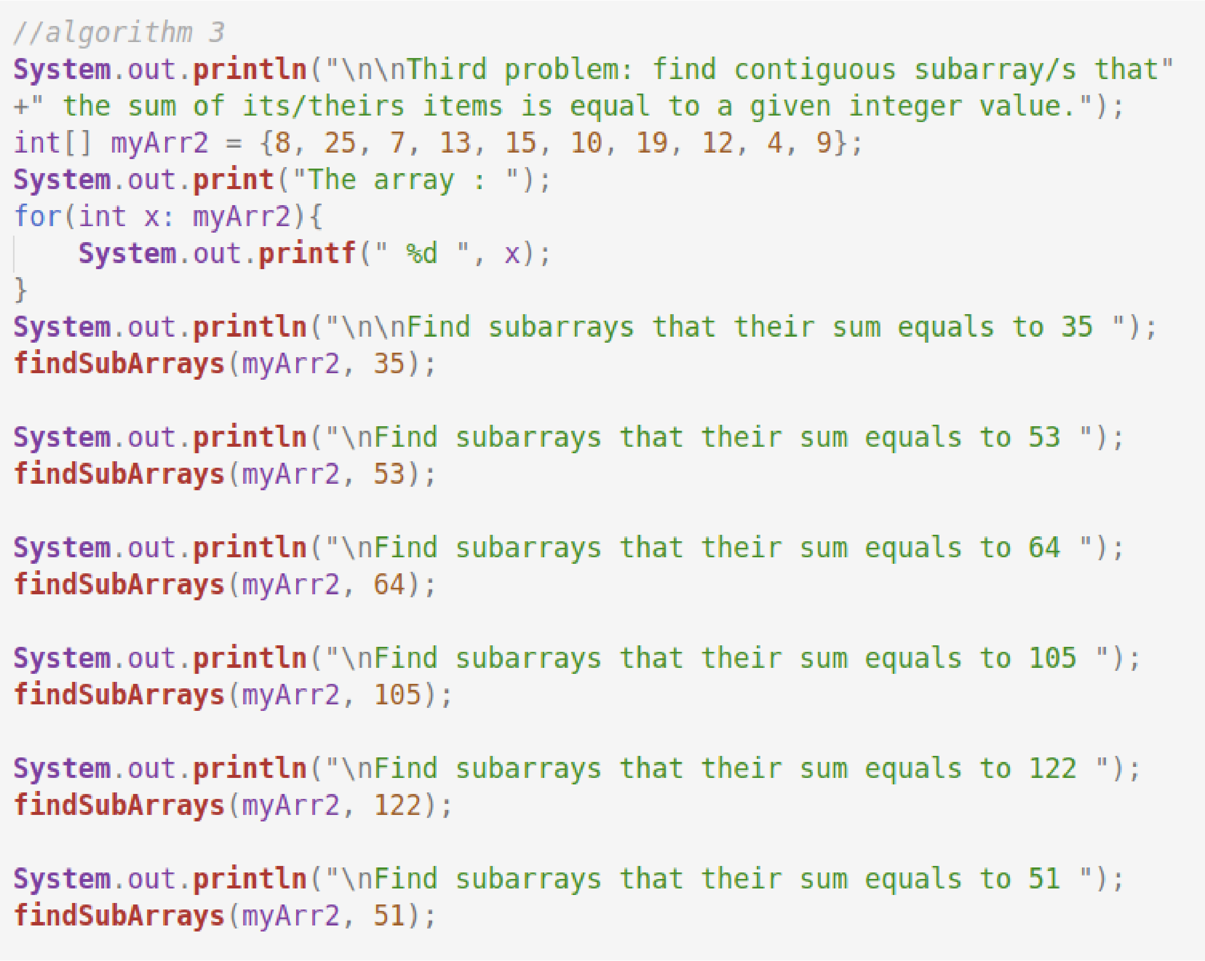
```java
String str = "askxymxynxysjkahxykasbhxy ";
String str2 = "xy";
System.out.println("First problem: Returns index of i th occurence of " + str2);
System.out.println("The string we will search is " + str);
for(int i = 0; i<= 5 ; i++){
    System.out.println("i = " + i);
    System.out.println("Answer = " + findString(str, str2, i));
}
```

2nd algorithm



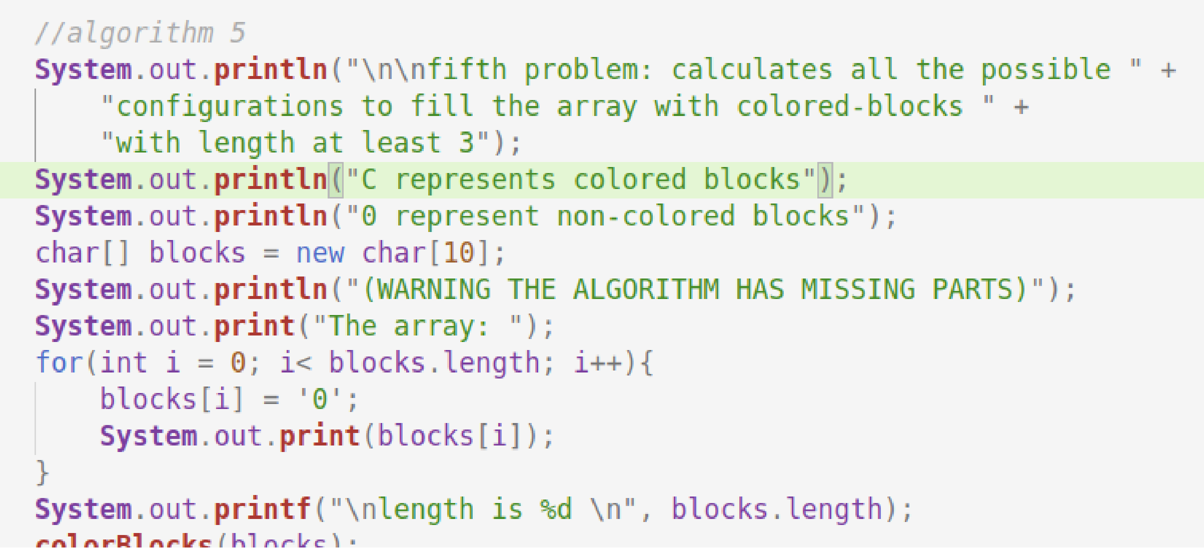
```java
//algorithm 2
System.out.println("\n\nSecond problem: Returns the number of"
    +" items between two number");
System.out.println("(It includes the borders)");
int[] myArr = {1, 2, 5, 8, 15, 16, 19, 23, 32, 35};
System.out.print("The array : ");
for(int x : myArr){
    System.out.printf(" %d ", x);
}
System.out.print("\n\nnumbers of items between 8 and 23 in the Array = ");
System.out.println(numberOfItemsBetween(myArr, 8, 23));
System.out.print("\n\n");
int l = 0 ,r;
for(int i = 1; i<38; i++){
    r = i;
    System.out.printf("Number of items between %d - %d", l, r);
    System.out.println(" = " + numberOfItemsBetween(myArr, l, r));
}
```

3rd algorithm



```
//algorithm 3
System.out.println("\n\nThird problem: find contiguous subarray/s that"
+" the sum of its/theirs items is equal to a given integer value.");
int[] myArr2 = {8, 25, 7, 13, 15, 10, 19, 12, 4, 9};
System.out.print("The array : ");
for(int x: myArr2){
    System.out.printf(" %d ", x);
}
System.out.println("\n\nFind subarrays that their sum equals to 35 ");
findSubArrays(myArr2, 35);

System.out.println("\nFind subarrays that their sum equals to 53 ");
findSubArrays(myArr2, 53);

System.out.println("\nFind subarrays that their sum equals to 64 ");
findSubArrays(myArr2, 64);

System.out.println("\nFind subarrays that their sum equals to 105 ");
findSubArrays(myArr2, 105);

System.out.println("\nFind subarrays that their sum equals to 122 ");
findSubArrays(myArr2, 122);

System.out.println("\nFind subarrays that their sum equals to 51 ");
findSubArrays(myArr2, 51);
```

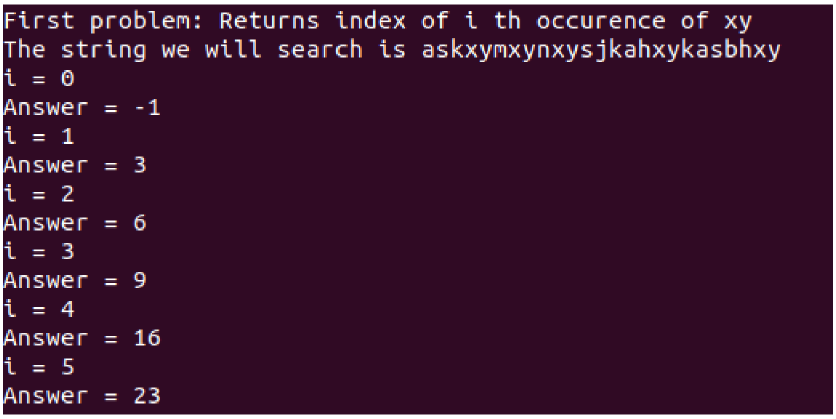
algorithm 5



```
//algorithm 5
System.out.println("\n\nfifth problem: calculates all the possible " +
    "configurations to fill the array with colored-blocks " +
    "with length at least 3");
System.out.println("C represents colored blocks");
System.out.println("0 represent non-colored blocks");
char[] blocks = new char[10];
System.out.println("(WARNING THE ALGORITHM HAS MISSING PARTS)");
System.out.print("The array: ");
for(int i = 0; i< blocks.length; i++){
    blocks[i] = '0';
    System.out.print(blocks[i]);
}
System.out.printf("\nlength is %d \n", blocks.length);
colorBlocks(blocks);
```

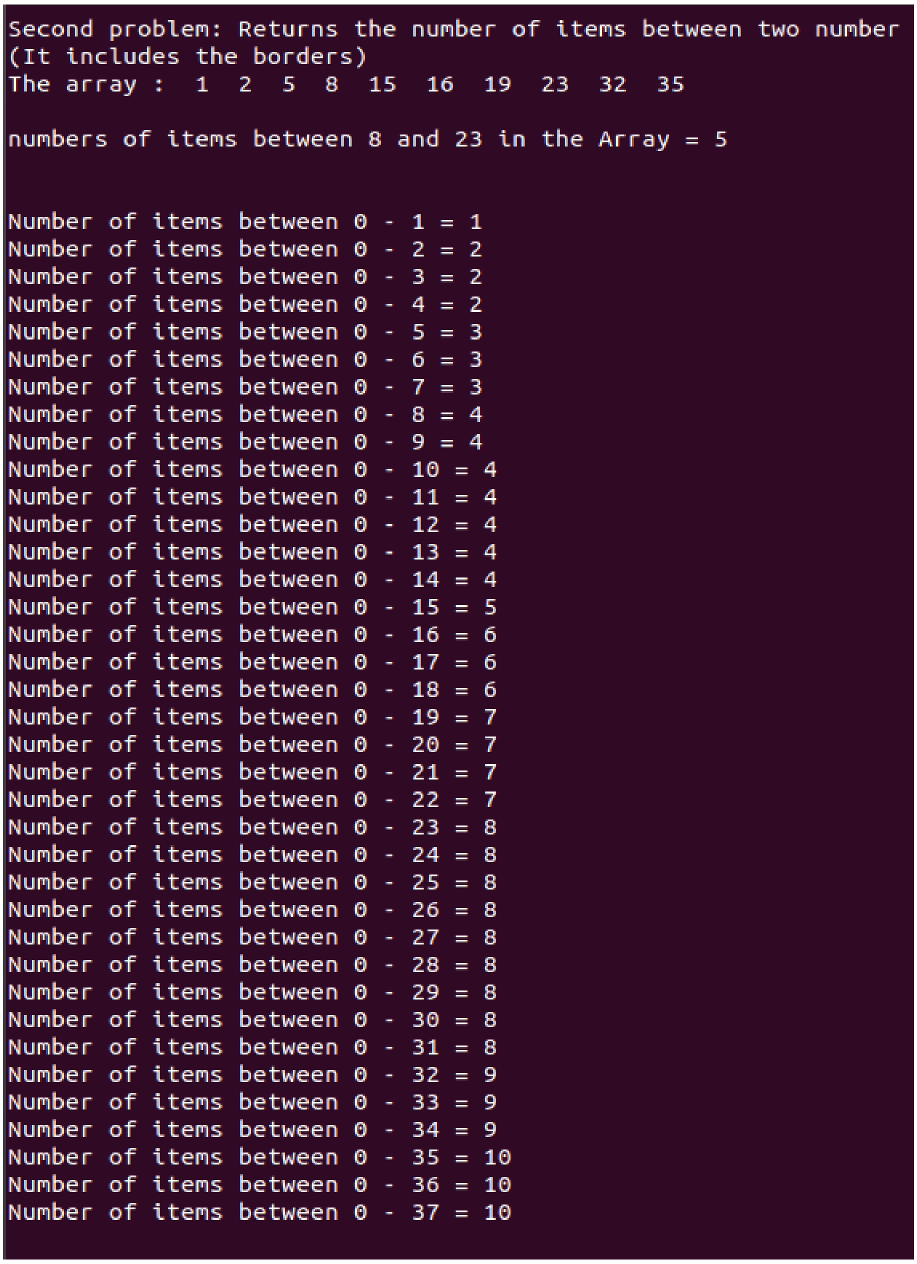
5- RUNNING AND RESULTS

Results for algorithm 1



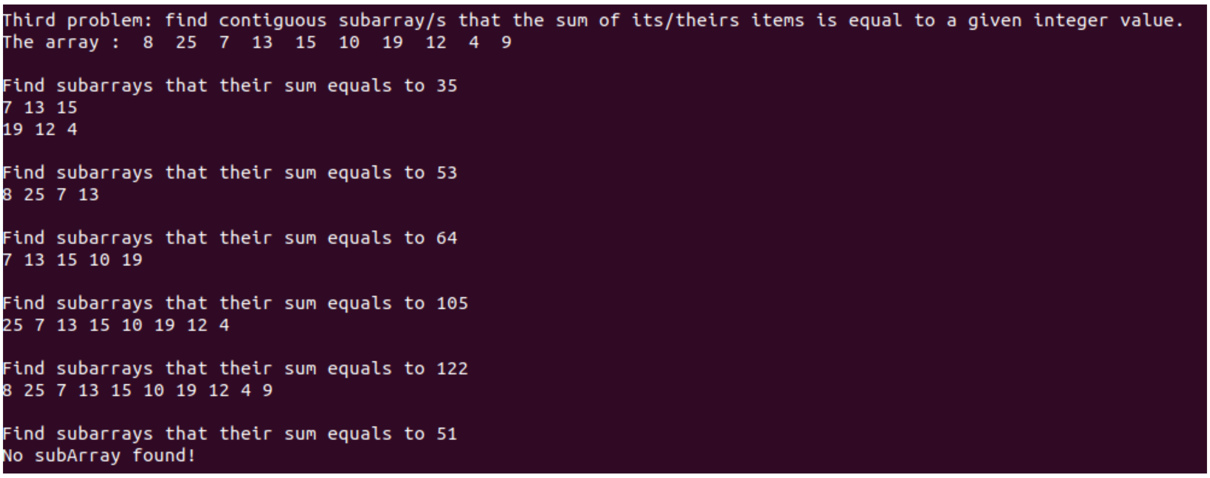
```
First problem: Returns index of i th occurence of xy
The string we will search is askxymxynxysjkahxykasbhxy
i = 0
Answer = -1
i = 1
Answer = 3
i = 2
Answer = 6
i = 3
Answer = 9
i = 4
Answer = 16
i = 5
Answer = 23
```

Results for algorithm 2



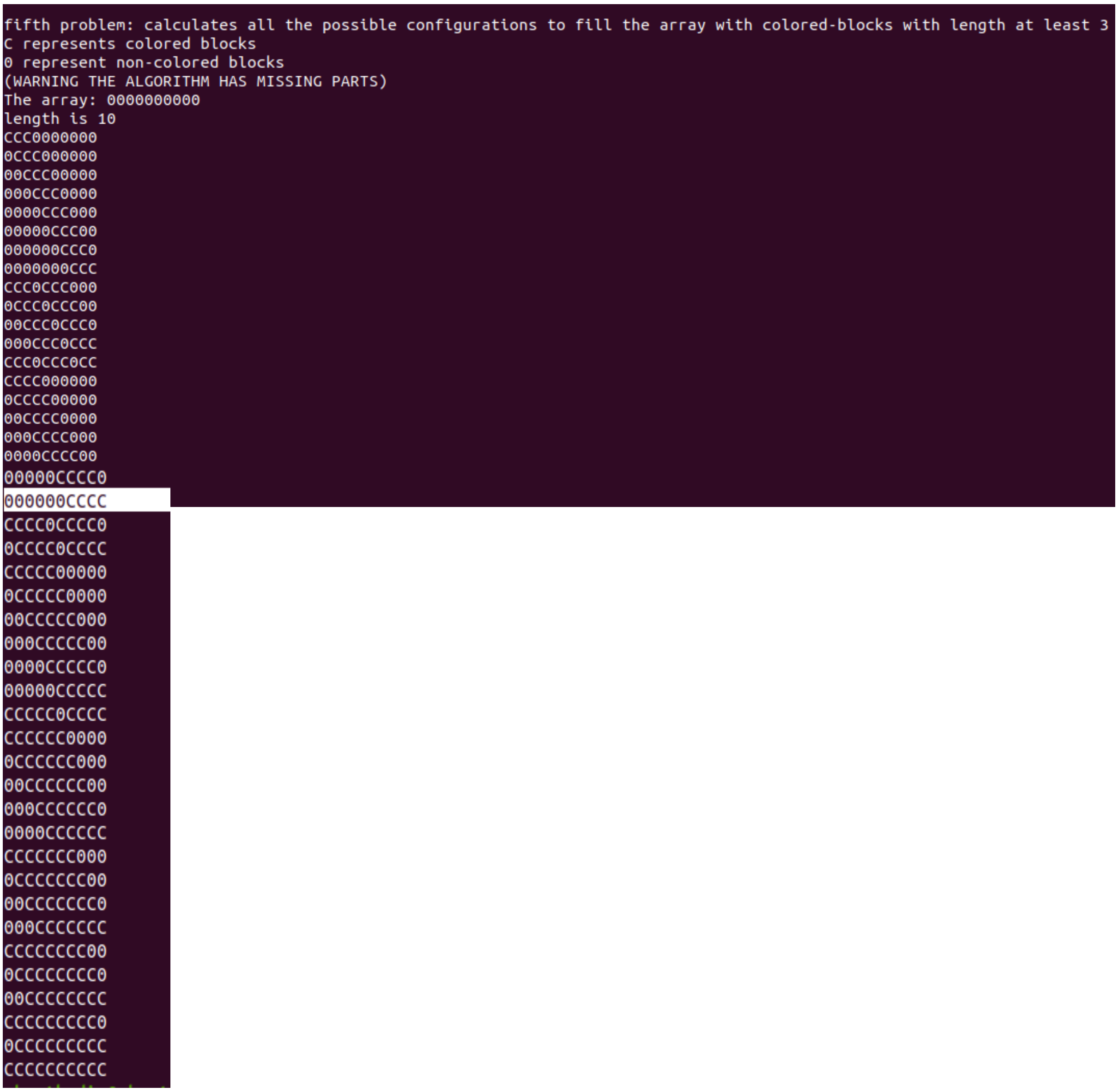
```
Second problem: Returns the number of items between two number
(It includes the borders)
The array :  1  2  5  8  15  16  19  23  32  35

numbers of items between 8 and 23 in the Array = 5


Number of items between 0 - 1 = 1
Number of items between 0 - 2 = 2
Number of items between 0 - 3 = 2
Number of items between 0 - 4 = 2
Number of items between 0 - 5 = 3
Number of items between 0 - 6 = 3
Number of items between 0 - 7 = 3
Number of items between 0 - 8 = 4
Number of items between 0 - 9 = 4
Number of items between 0 - 10 = 4
Number of items between 0 - 11 = 4
Number of items between 0 - 12 = 4
Number of items between 0 - 13 = 4
Number of items between 0 - 14 = 4
Number of items between 0 - 15 = 5
Number of items between 0 - 16 = 6
Number of items between 0 - 17 = 6
Number of items between 0 - 18 = 6
Number of items between 0 - 19 = 7
Number of items between 0 - 20 = 7
Number of items between 0 - 21 = 7
Number of items between 0 - 22 = 7
Number of items between 0 - 23 = 8
Number of items between 0 - 24 = 8
Number of items between 0 - 25 = 8
Number of items between 0 - 26 = 8
Number of items between 0 - 27 = 8
Number of items between 0 - 28 = 8
Number of items between 0 - 29 = 8
Number of items between 0 - 30 = 8
Number of items between 0 - 31 = 8
Number of items between 0 - 32 = 9
Number of items between 0 - 33 = 9
Number of items between 0 - 34 = 9
Number of items between 0 - 35 = 10
Number of items between 0 - 36 = 10
Number of items between 0 - 37 = 10
```

Results for algorithm 3



```
Third problem: find contiguous subarray/s that the sum of its/theirs items is equal to a given integer value.
The array :  8  25  7  13  15  10  19  12  4  9

Find subarrays that their sum equals to 35
7 13 15
19 12 4

Find subarrays that their sum equals to 53
8 25 7 13

Find subarrays that their sum equals to 64
7 13 15 10 19

Find subarrays that their sum equals to 105
25 7 13 15 10 19 12 4

Find subarrays that their sum equals to 122
8 25 7 13 15 10 19 12 4 9

Find subarrays that their sum equals to 51
No subArray found!
```

Results for algorithm 5



```
fifth problem: calculates all the possible configurations to fill the array with colored-blocks with length at least 3
C represents colored blocks
0 represent non-colored blocks
(WARNING THE ALGORITHM HAS MISSING PARTS)
The array: 0000000000
length is 10
CCC0000000
0CCC000000
00CCC00000
000CCC0000
0000CCC000
00000CCC00
000000CCC0
0000000CCC
CCC0CCC000
0CCC0CCC00
00CCC0CCC0
000CCC0CCC
CCC0CCC0CC
CCCC000000
0CCCC00000
00CCCC0000
000CCCC000
0000CCCC00
00000CCCC0
000000CCCC
CCCC0CCCC0
0CCCC0CCCC
CCCCC00000
0CCCCC0000
00CCCCC000
000CCCCC00
0000CCCCC0
00000CCCCC
CCCCC0CCCC
CCCCCC0000
0CCCCCC000
00CCCCCC00
000CCCCCC0
0000CCCCCC
CCCCCCC000
0CCCCCCC00
00CCCCCCC0
000CCCCCCC
CCCCCCCC00
0CCCCCCCC0
00CCCCCCCC
CCCCCCCCC0
0CCCCCCCCC
CCCCCCCCCC
```