GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 4
Due date: March 27, 2022– 23:55

This homework aims to improve your recursive solution development ability for concordant problems. **Design** your solutions with respect to the given restrictions in the questions and do the **implementations** of these algorithms in Java.

Give the time **complexity analysis** of the solution algorithms for Q1, Q2, Q3, and Q4 in the report **in detail**. Analyze the correctness of your solution for at least two questions by using the induction method.

Q1. Write a recursive function to search a given string in another given bigger string. The function should return the index of the i^{th} occurrence of the query string and return -1 when the query string doesn't occur in the big string or the number of occurences is less than *i*.

Q2. Suppose that you are given a sorted integer array. Suggest a recursive algorithm to find the number of items in the array between two given integer values. Hint: Consider an approach like binary search algorithm; compare given two integers with the middle element.

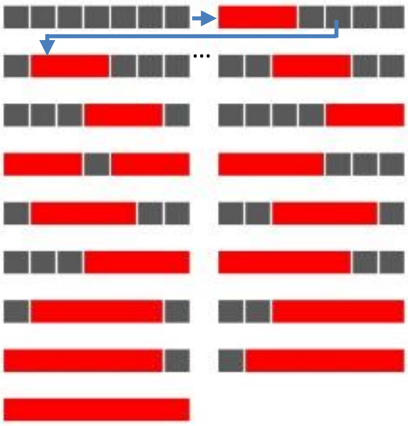
Q3. Suppose that you are given an unsorted integer array. Propose a recursive solution to find contiguous subarray/s that the sum of its/theirs items is equal to a given integer value. Hint: Consider the following approach; the first element can be a part of the contiguous subarray or not. You could perform two recursive calls based on these two cases.

Q4. Explain the output of the given recursive function and how it works in detail. Analyze the time complexity of this function by analyzing the number of multiplication operations (which is the basic operation) at the base case.

```
# foo (integer1, integer2)

    if (integer1 < 10) or (integer2 < 10)
      return integer1 * integer2

    //number_of_digit returns the number of digits in an integer
    n = max(number_of_digits(integer1), number_of_digits(integer2))
    half = int(n/2)

    // 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
    int1, int2 = split_integer (integer1, half)
    int3, int4 = split_integer (integer2, half)

    sub0 = foo (int2, int4)
    sub1 = foo ((int2 + int1), (int4 + int3))
    sub2 = foo (int1, int3)

    return (sub2*10^(2*half))+((sub1-sub2-sub0)*10^(half))+(sub0)
```

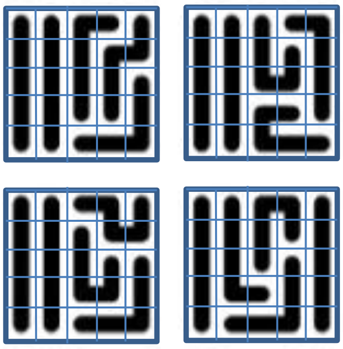
The following two questions worth extra 100 points.

Q5. Suppose that you are given a 1-D array of empty blocks of length *L*. Propose a recursive algorithm that calculates all the possible configurations to fill this array with colored-blocks with length at least 3 (i. e., the length of colored blocks can be from 3 to *L*). Note that there should be at least one empty block between each consecutive colored-block.



Print all the steps of your algorithm as shown in the example above.

Q6. Consider a problem similar to the problem in Q5 which is defined on two-dimensional space. Given an *n* x *n* matrix of empty blocks, propose a recursive solution that calculates all the possible combinations to fill this matrix by using snakes of length exactly *n*. Please note that snakes can bend as shown in the example below. In this example, the size of the



matrix (*n*) is 5x5 and the matrix is filled by using the snakes of length 5. Print all the possible configurations for filling the matrix.

RESTRICTIONS:

- You can use all the abstract data structures you have learned in the class so far.
- You can use third-party libraries for the printings in Q1 and Q2.

GENERAL RULES:

- For any question firstly use the forum on the MS Teams page, and then the contact TA.
- You can submit an assignment one day late and will be evaluated over sixty percent (%60).

TECHNICAL RULES:

- You must write a driver function that demonstrates all possible actions in your homework. For example, if you are asked to implement an array list and perform an iterative search on the list then, you must at least provide the following in the driver function:
    - Create an array list and add items to the list. Append items to the head, tail, and *k*th index of the list.
    - Perform at least two different searches by using two items in the list and print the index of the items.
    - Perform another search with an item that isn't in the array list and inform the user that the item doesn't exist in the array list.
    - Delete an existing item from the list and repeat the searches.
    - Try to delete an item that is not on the array list and throw an exception for this situation.

  The driver function should run when the code file is executed.

- Implement clean code standards in your code;
    - Classes, methods, and variables names must be meaningful and related to the functionality.
    - Your functions and classes must be simple, general, reusable, and focus on one topic.
    - Use standard java code name conventions.

REPORT RULES:

- Add all javadoc documentations for classes, methods, variables ...etc. All explanations must be meaningful and understandable.
- You should submit your homework code, Javadoc, and report to MS Teams in a "studentid_hw1.tar.gz" file.
- Use the given homework format including selected parts from the table below:

Detailed system requirements	✓
The Project use case diagrams	X
Class diagrams	✓

Other diagrams (extra points)	X
Problem solutions approach	✓
Test cases	✓
Running command and results	✓

GRADING :

No OOP design:	**-100**
No error handling:	**-50**
No javadoc documentation:	-50
No report:	-90
Disobey restrictions:	-100
Cheating:	**-200**
Your solution is evaluated over 100 as your performance.	

CONTACT :

- Teaching Assistant M. Burak Koca, b.koca@gtu.edu.tr