Assignment 1

Deadline: Sunday, September 20, 23:59.

Instruction

This assignment is backed by Chapter 13. You have the maximum control of the way you program, but:

- Your code should be well-written in terms of the design (careful considerations and bug-freeness) and efficiency (few duplicates).
- The program must be adequately commented and follow a coding style. See the "Introductory video" on Canvas to find references for basic rules of writing comments and programming style.

Question 1

Use recursion to implement a method

```
public static int indexOf(String text, String str)
```

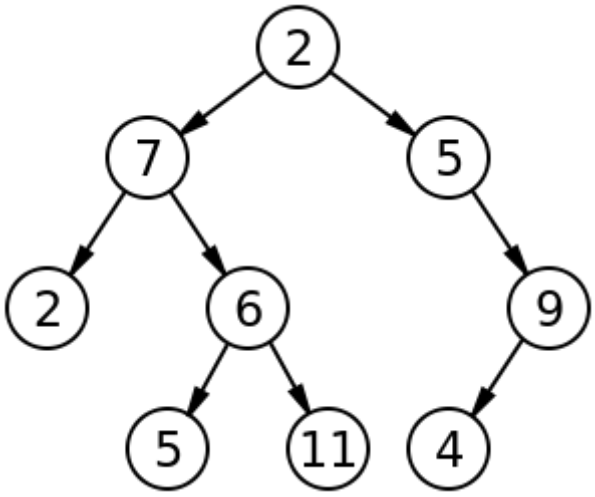
that returns the starting position of the first substring of the `text` that matches `str`. Return -1 if `str` is not in `text`. For instance, `s.indexOf("Mississippi","sip")` gives `6`.

Question 2

Given a set represented as string, write a recursive program to print all subsets of it. The subsets can be printed in any order. For instance, the input string “abc” generates the following subsets: "", "a", "b", "c", "ab", "ac", "bc", "abc". Note that “” stands for the “null string”.

Question 3

Binary tree is a basic concept in data structure. Here is a diagram of a binary tree.



As you can see, each node here contains some data, and has maximum 2 child nodes. For instance, the node with data 6 has two nodes, respectively having data 5 and 11. In addition, this particular tree is of height 3 (the root does not count towards the height). Now, write a class `Tree` to represent a binary tree of a general height. You can put anything you like as the actual data in the node. Write a program to print your tree on the screen like the following.

```
A
|----B
|----|----C
|----|----|----D
|----|----|----E
|----|----F
|----|----|----G
|----|----|----H
|----I
|----|----J
|----|----|----K
|----|----|----L
|----|----M
|----|----|----N
|----|----|----O
```

where A, B, C, etc. are some data. In this example, A is the data for the root node of the tree and B and I represent the data for the left child and the right child of the root node, respectively. Likewise, C is the data for the left child of the node with data B and F is the data for the right child of the node with data B and so on.

The main method that does the job of printing the tree must be a recursive method.

Question 4

You are now in a maze. The walls of the maze are indicated by asterisks(`*`)

```
*x*******
*     * *
* ***** *
* * *   *
* * *** *
*   *   *
*** * * *
*     * *
*******x*
```

The locations marked with `x` are exit/entrance. Write a recursive program to help you escape from the maze, starting from one of the entrance points. You might have expected this: you can not exit from the entrance where you started. There are many ways to write the program (see also exercise P13.18). What I did was to visit every path recursively until I see one leading to an exit. Note that to solve this question you need to construct the maze yourself. Any maze such as the one illustrated above can be represented by a 2x2 matrix. Also note that your program must solve any maze problem passed as an input.

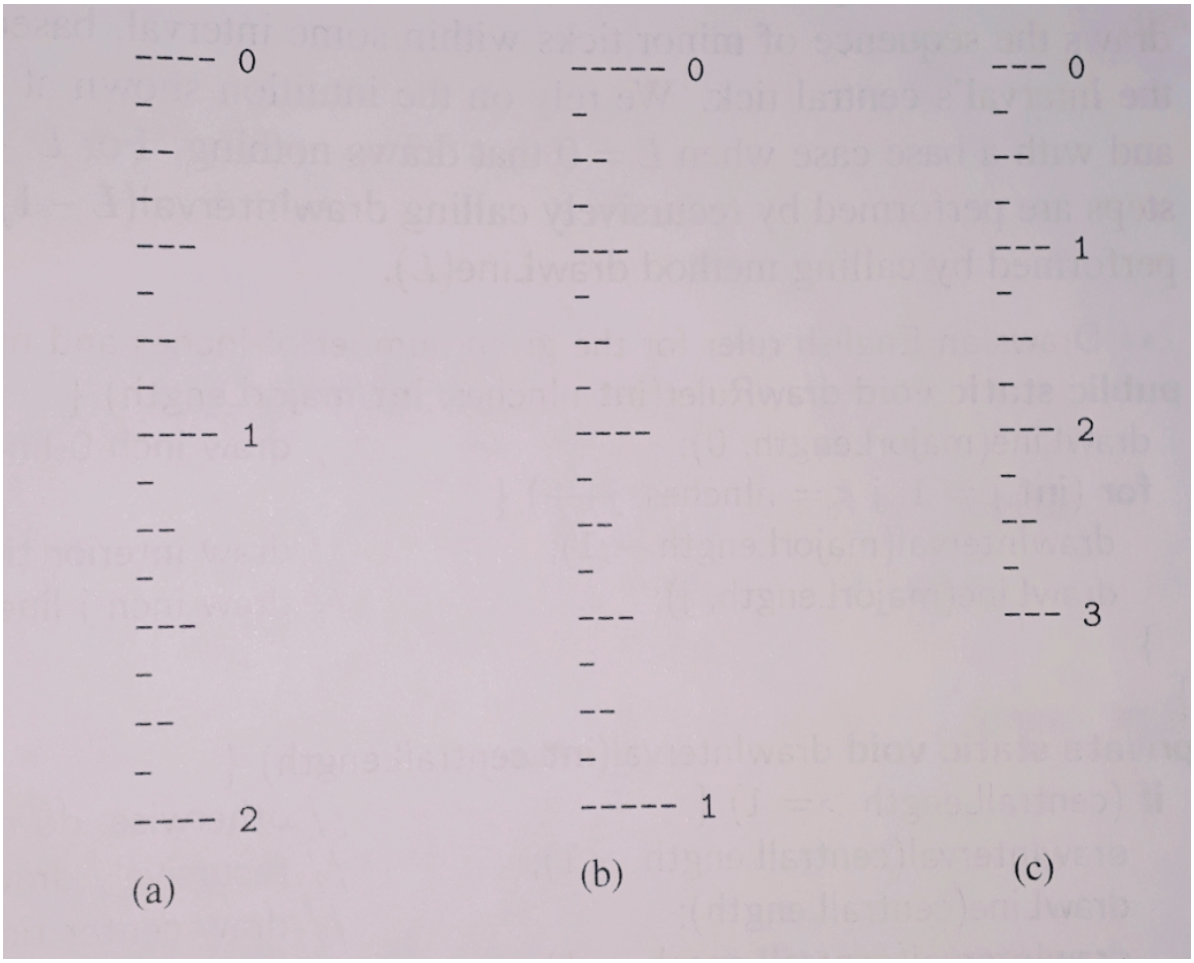
Your program must also print out the route like the following:

```
*4*******
*4    * *
*4***** *
*4* *   *
*4* *** *
*114*114*
***4*2*4*
*  112*4*
*******x*
```

Where 1 represents a rightward move, 2 an upward move, 3 a leftward move and 4 a downward move indicating the steps taken from the entrance to the exit.

Question 5

In this assignment we will consider how to draw the markings of a typical English ruler. For each inch, we place a tick with a numeric label. We denote the length of the tick designating a whole inch as the **major tick length.** Between the marks for whole inches, the ruler contains a series of **minor ticks,** placed at intervals of ½ inch, ¼ inch, and so on. As the size of the intervals decreases by half, the tick length decreases by one. The Figure 1 demonstrates several such rulers with varying major tick lengths (although not drawn to scale)



Three sample outputs of an English ruler drawing are shown in the above figure: (a) a 2-inch ruler with major tick length 4; (b) a 1-inch ruler with major tick length 5; (c) a 3-inch ruler with major tick length 3.

The English ruler pattern has a self-recursive structure shape at various levels of magnification. Write a program that contains a recursive implementation of a method that draws a ruler. The main method of your program, *drawRuler*, manages the construction of the entire ruler. Its arguments specify the total number of inches in the ruler and the major tick length, as follows:

```
public static void drawRuler(int nInches, int majorLength)
```

Question 6

A common application of backtracking is when we want to enumerate various configurations in order to solve a combinatorial puzzle. For example, the following are three instances of what are known as summation puzzles:

```
pot + pan = bib
dog + cat = pig
boy + girl = baby
```

To solve such a puzzle, we need to assign a unique digit (that is, 0, 1, ..., 9) to each character in the equation, in order to make the equation true. Using backtracking, write a program for solving summation puzzles by enumerating and testing all possible configurations. To be clear, the input for your program is of the form `A + B = C`, where A, B and C are sequences of characters (string). Note that your program should be generic and solve any summation puzzle problem passed as an input. If a summation puzzle problem has no solution, your program should indicate that as return. Also note that different characters can be assigned to the same digit, although identical characters (from the summation puzzle) must be assigned to the same digit.

Using your program, solve the three puzzles given above.