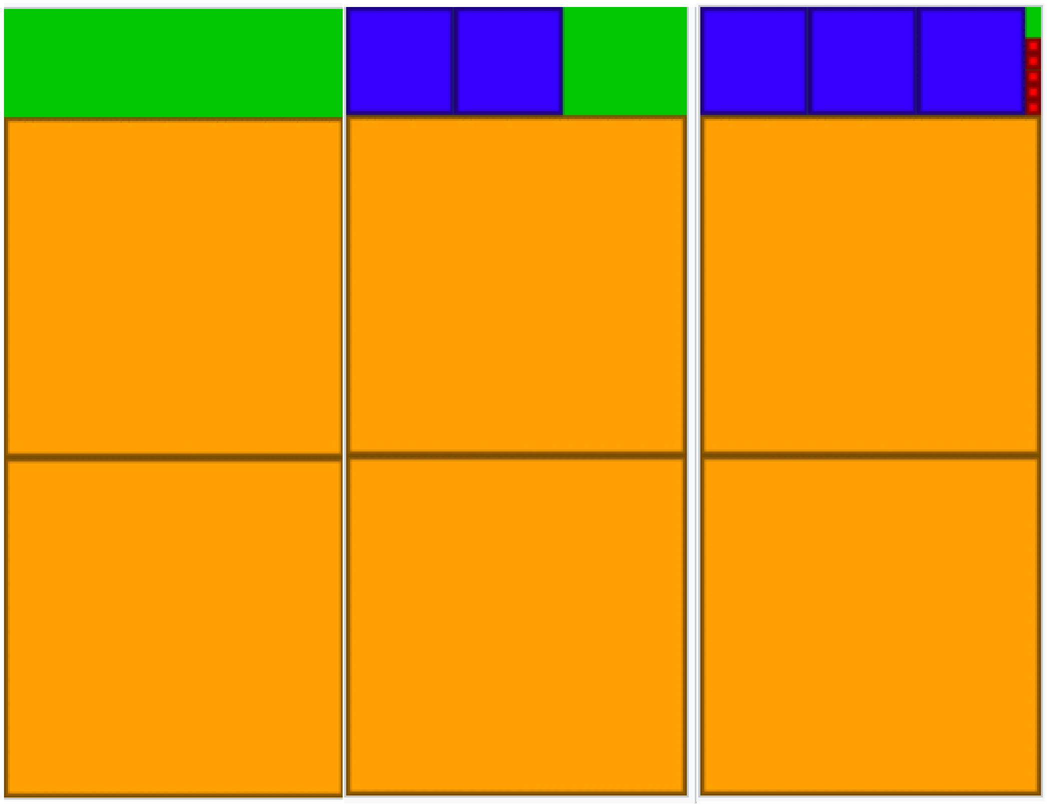
Sofia University
Department of Mathematics and Informatics

Course : Applied OO Programming part 1
Date: March 30, 2020
Student Name:

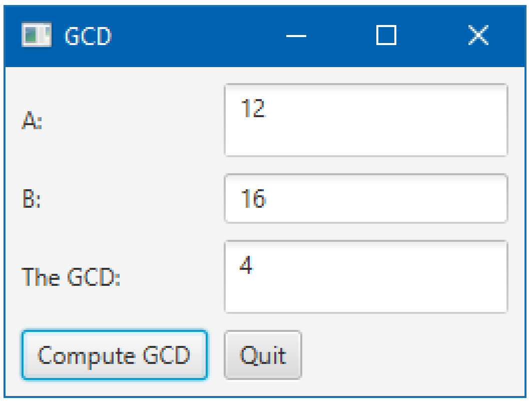
Lab No. 6

Problem 1a

The Euclidean algorithm for finding **the Greatest Common Divisor** can be visualized as follows. Assume that we wish to cover an *a*-by-*b* rectangle with square tiles exactly, where *a* is the larger of the two numbers. We first attempt to tile the rectangle using *b*-by-*b* square tiles; however, this leaves an *(a mod b)*-by-*b* residual rectangle untiled. We then attempt to tile the residual rectangle with *(a mod b)*-by- *(a mod b)* square tiles. This leaves a second residual rectangle [*(a mod b) mod b*]--by-*(a mod b)*-, which we attempt to tile using [*(a mod b) mod b*]-by-[*(a mod b) mod b*] square tiles, and so on. The sequence ends when there is no residual rectangle, i.e., when the square tiles cover the previous residual rectangle exactly. The length of the sides of the smallest square tile is the GCD of the dimensions of the original rectangle. For example, the smallest square tile in the adjacent figure is 21-by-21 (shown in red), and 21 is the GCD of 1071 and 462, the dimensions of the original rectangle (shown in green).



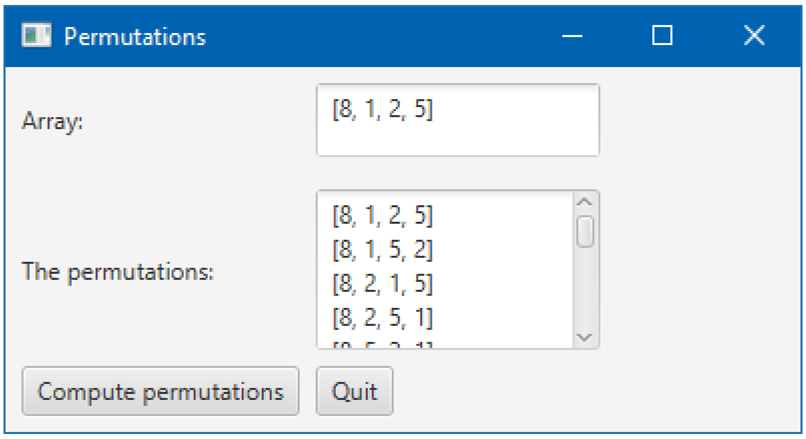
Write a recursive implementation of the GCD algorithm and **demonstrate its execution in a JavaFXML application**. The JavaFXML application allows the user to **input two integers** and has a button to compute and display the GCD.



Problem 2a

Consider the problem of **computing all the permutations of the elements of an array**. For example, if the array is <2,3,5, 7> one output could be <2,3,5, 7>, <2,3, 7,5>, <2,5,3, 7>, <2,5, 7,3>, <2,7,3,5>, <2,7,5,3>, <3,2,5,7>, <3,2,7,5>, <3,5,2,7>, <3,5,7,2>, <3,7,2,5>, <3,7,5,2>, <5, 2,3, 7>,<5, 2, 7,3>, <5,3,2,7>, <5,3,7,2>, <5,7,2,3>, <5,7,2,3>, <7,2,3,5>, <7,2,5,3>, <7,3, 2,5>, <7,3,5, 2>, <7,5, 2,3>, <7,5,3, 2>. (Any other ordering is acceptable too.)

Write a JavaFXML application program which takes as input an array of 4 random generated distinct single-digit integers and use a recursive method to generate all the permutations of the elements in that array. No permutation of the array may appear more than once

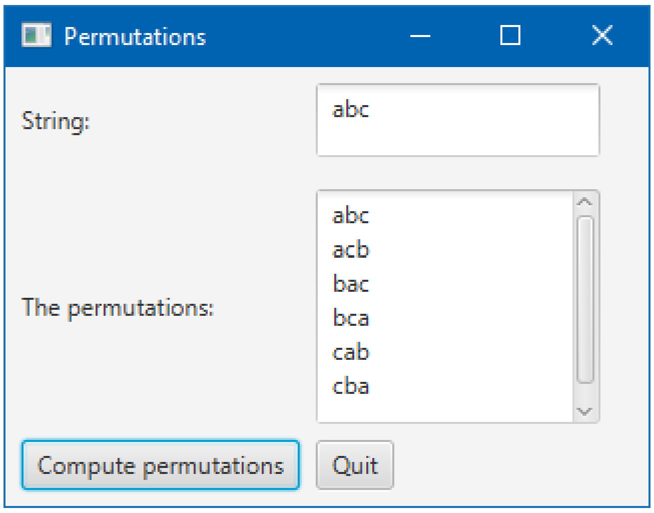


Problem 2a

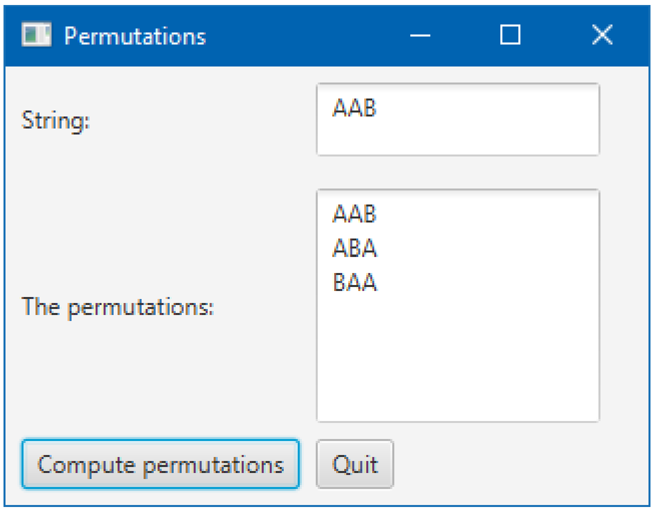
Solve Problem 2a when the input array may have duplicates.
You should not repeat any permutations. For example, if A = <2, 2,3, 0> then the output should be <2, 2,0,3>, <2, 2, 3, 0>, <2,0, 2,3>, <2, 0,3, 2>, <2,3, 2, 0>, <2,3, 0, 2>, <0, 2, 2,3>, <0, 2,3, 2>, <0,3, 2, 2>, <3, 2, 2,0>, <3, 2, 0, 2>, <3, 0, 2, 2>

Problem 3

Write a recursive method to **print all permutations of String**. The method must take care of duplicate characters. For example “AAB” permutations will be “AAB”, “ABA” and “BAA”, while for "abc" the permutations should be "abc", "acb" "bac" "bca" "cab" "cba". Implement the method in a JavaFXML application, where the user inputs a String in a textbox and displays all the permutations of its characters in a text area.
Hint. Convert String to char[] and use operations with arrays.



```
abc
acb
bac
bca
cab
cba
```



```
AAB
ABA
BAA
```

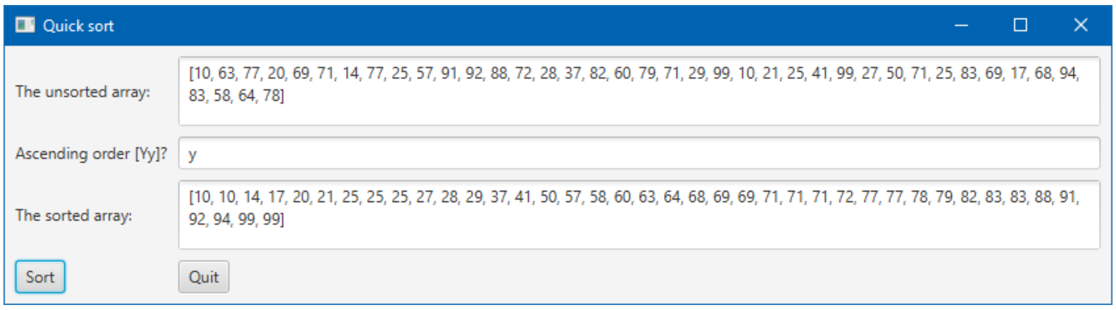
Problem 4

Implement the **Quick sort algorithm** in Java allowing to sort an `ArrayList` of `Integer` numbers in **ascending** and **descending** order. The method implementing the algorithm should be in the form:

```
public ArrayList<Integer> sort(ArrayList <Integer> listToSort)
```

The method sorts `listToSort` in ascending or descending order a datamember `sortOrder` is correspondingly set to `true` or `false`.

Implement the method in a JavaFXML application with the following GUI

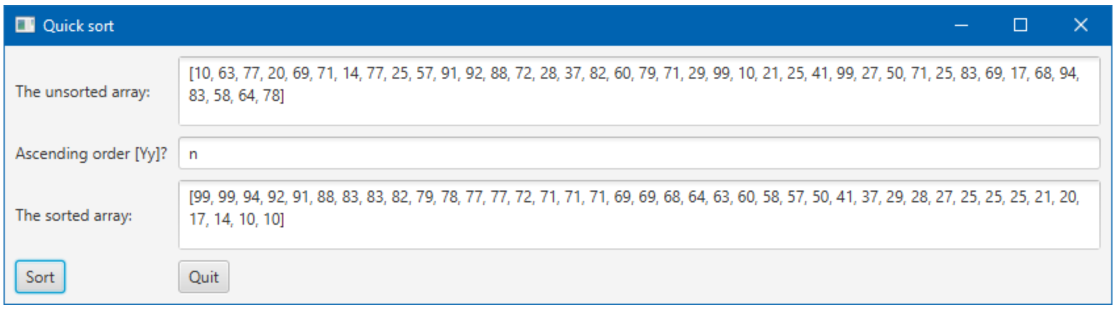


Quick sort

The unsorted array: [10, 63, 77, 20, 69, 71, 14, 77, 25, 57, 91, 92, 88, 72, 28, 37, 82, 60, 79, 71, 29, 99, 10, 21, 25, 41, 99, 27, 50, 71, 25, 83, 69, 17, 68, 94, 83, 58, 64, 78]

Ascending order [Yy]? y

The sorted array: [10, 10, 14, 17, 20, 21, 25, 25, 25, 27, 28, 29, 37, 41, 50, 57, 58, 60, 63, 64, 68, 69, 69, 71, 71, 71, 72, 77, 77, 78, 79, 82, 83, 83, 88, 91, 92, 94, 99, 99]



Quick sort

The unsorted array: [10, 63, 77, 20, 69, 71, 14, 77, 25, 57, 91, 92, 88, 72, 28, 37, 82, 60, 79, 71, 29, 99, 10, 21, 25, 41, 99, 27, 50, 71, 25, 83, 69, 17, 68, 94, 83, 58, 64, 78]

Ascending order [Yy]? n

The sorted array: [99, 99, 94, 92, 91, 88, 83, 83, 82, 79, 78, 77, 77, 72, 71, 71, 71, 69, 69, 68, 64, 63, 60, 58, 57, 50, 41, 37, 29, 28, 27, 25, 25, 25, 21, 20, 17, 14, 10, 10]

Use `class Random` to generate the elements of `listToSort`

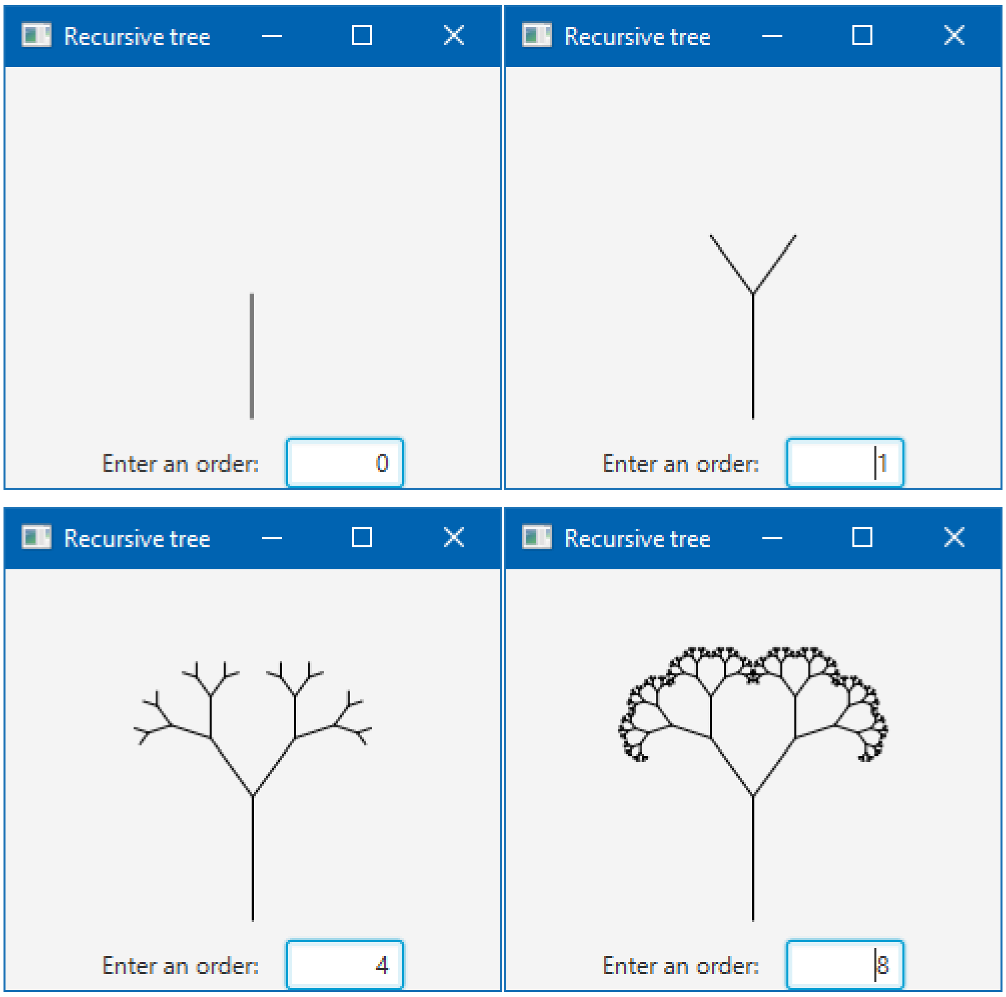
Problem 5

Write a **recursive method** to compute all size `k` subsets of `{1, 2,..., n}`, where `k` and `n` are program inputs i.e. all the combinations of size k from n numbers. For example, if `k = 2` and `n = 5`, then the result is the following:

`((1, 2), (1,3), (1, 4), (1,5), (2,3), (2, 4), (2,5), (3, 4), (3,5), (4, 5))`. The method returns an `ArrayList` of the subsets, where each subset is represented by an `ArrayList`.

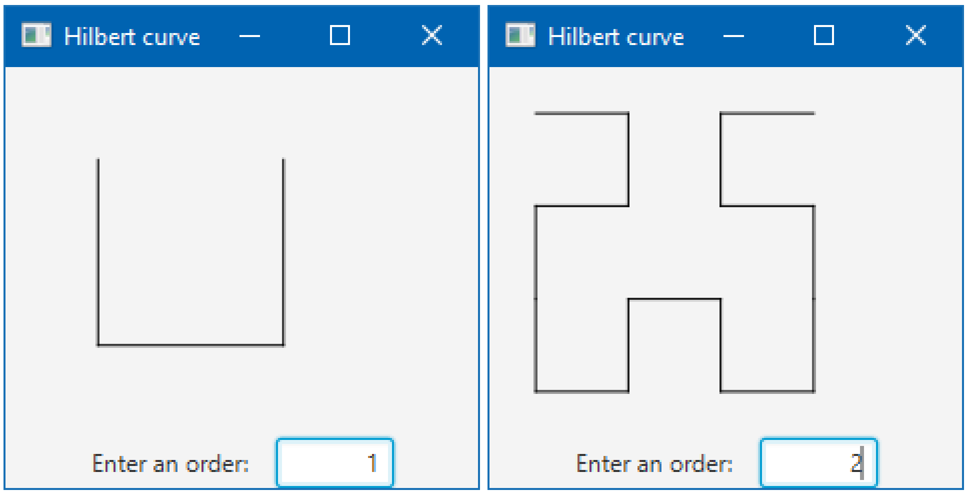
Problem 6

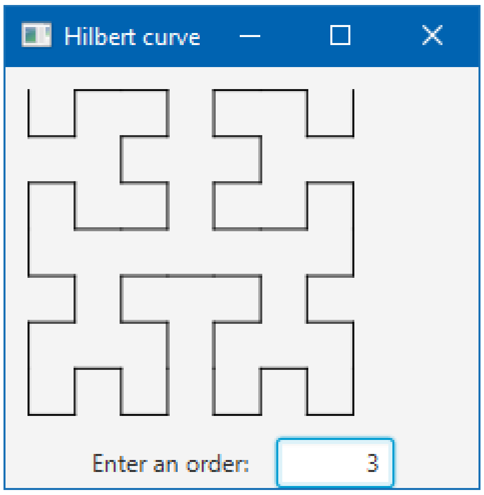
Write a JavaFX application program to display a recursive tree as shown below:

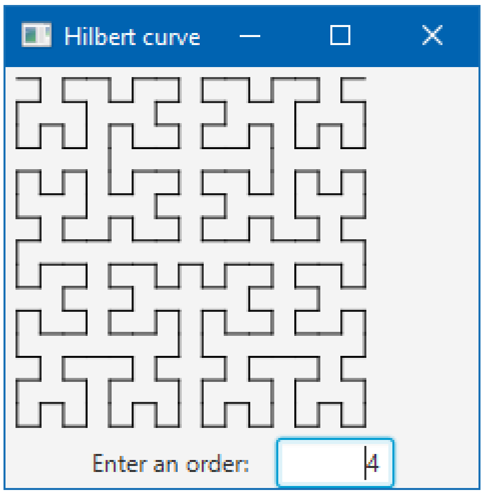


Problem 7

The Hilbert curve, first described by German mathematician David Hilbert in 1891, is a space-filling curve that visits every point in a square grid with a size of 2 * 2, 4 * 4, 8 * 8, 16 * 16, or any other power of 2. Write a program that displays a Hilbert curve for the specified order, as shown below

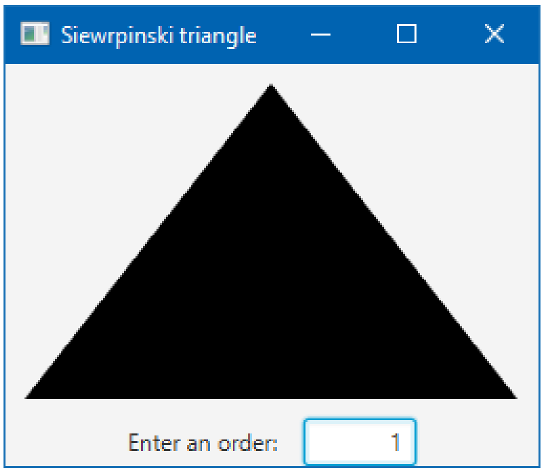


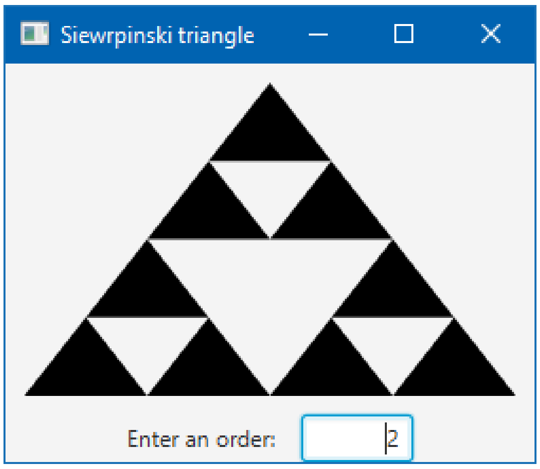


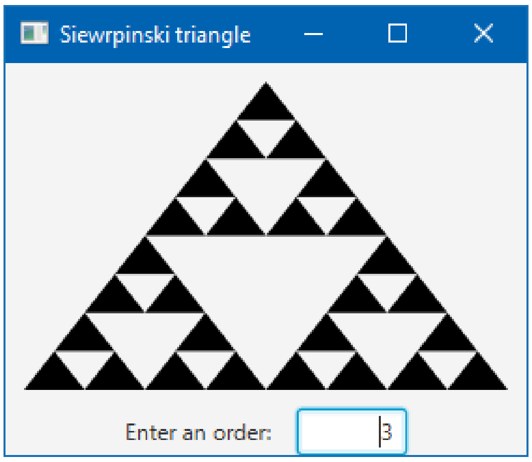


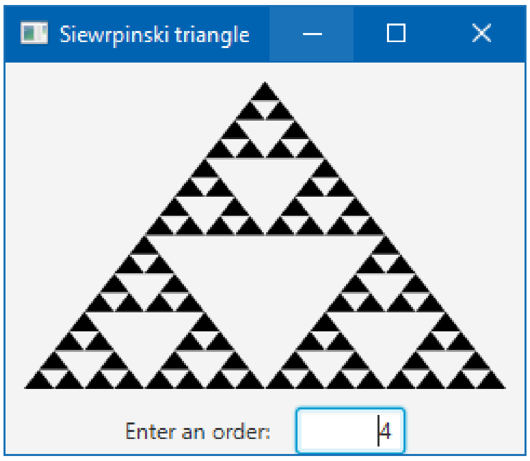
Problem 8

Write a program that lets the user to enter the order and display the filled Sierpinski triangles as shown below:









Problem 9

Write a recursive method to compute the following series:

$$\frac{1}{3} + \frac{2}{5} + \frac{3}{7} + \frac{4}{9} + \frac{5}{11} + \frac{6}{13} + \ldots + \frac{i}{2i + 1}$$

Write a test program that displays the partial sums for i = 1, 2, . . ., 10.

Problem 10

Write **a JavaFX application,** which **computes** the **one's and the two's complement** of a given decimal number. The **number is input as a decimal number in a text box**, **the one's and the two's complement are initiated by means of user defined buttons** and the **result (displayed in binary digits) of each computation** should be displayed in separate textboxes (with disabled editing property). For validation purposes, **there should be a textbox** displaying the sum of the obtained **one's and the two's compliment of the given number.**

Problem No. 11

You normally use an ***int*** to hold an integer value. Internally an ***int*** stores its value as a sequence of ***32*** bits, where each ***bit*** can be either ***0*** or ***1***. Most of the time you don't care about this internal binary representation; you just use an ***int*** type as a bucket to hold an ***integer*** value. In other words, occasionally a program might use an ***int*** because it holds ***32*** bits and not because it can represent an ***integer***.

We'd like to use an ***int*** not as an ***int*** but as an ***array*** of ***32 bits***. Therefore, the best way to solve this problem is to use an ***int*** as if it were an array of ***32 bits***! In other words, if bits is an ***int***, what we'd like to be able to write to access the bit at ***index 6*** is:

bits.get(6)

And, for example, set the bit at ***index 6*** to ***true***, we'd like to be able to write:

bits.set(6, true)

Write a ***class IntBits*** , allowing you the following operations

```
int adapted = 63;
IntBits bits = new IntBits(adapted);
boolean peek = bits.get(6); // retrieve boolean at index 6
bits.set(0, true); // set the bit at index 0 to true
bits.set(31, false); // set the bit at index
```

Problem No. 12

Write a program that **reverses** the order of the bits in an integer value. The program inputs an integer value and calls a method `reverseBits`() to print the reverse of the bits. Print the value of the integer before its bits have been reversed and after that. Consider both an **iterative** and **recursive** solution.