# MP2, Recursion

1. Implement class StringPermutation that prints all the permutations of a string. : Define the following two methods. The second is a helper method.)

   public static void displayPermutation(String s)
   public static void displayPermutation(String s1, String s2)

   The first method simply invokes displayPermutation(" ", s). The second method uses a loop to move a character from s2 to s1 and **recursively** invokes it with a new s1 and s2. The base case is that s2 is empty and prints s1 to the console.

```java
package r;
import java.util.Scanner;
public class StringPermutation
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String s = input.nextLine();
        System.out.println("The permutation for " + s + " is");
        displayPermuation(s);
    }
    public static void displayPermuation(String s)
    {...3 lines }
    public static void displayPermuation(String s1, String s2)
    {...14 lines }
}
```

Output

Run (MP1-F22-1.0) | Debugger Console | recursion (run)

```
run:
Enter a string: abc
The permutation for abc is
abc
acb
bac
bca
cab
cba
BUILD SUCCESSFUL (total time: 5 seconds)
```

2. Implement class DecimalToBinary that converts a decimal number into a binary number as a string recursively.

```java
package r;
import ...
public class DecimalToBinary
{
  public static void main(String[] args) {
    while ( true )
      {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter a decimal integer: ");
      int decimal = input.nextInt();
      System.out.println(decimal + " is binary " + decimalToBinary(decimal));
      }
  }
  public static String decimalToBinary(int value)
  {...6 lines }
}
```

Output

Run (MP1-F22-1.0) | Debugger Console | recursion (run) | recursion (run) #2 | recursion (run) #3 | recursion

```
run:
Enter a decimal integer: 12
12 is binary 1100
Enter a decimal integer: 123
123 is binary 1111011
Enter a decimal integer: 0
0 is binary
Enter a decimal integer: -1
-1 is binary -1
Enter a decimal integer:
```

3. Implement class BinaryToDecimal that parses a binary number as a string into a decimal integer, recursively.
The low and high parameters are used to denote the beginning and ending index of the string parsed.

```java
public class BinaryToDecimal
{public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        while (true){
            System.out.print("Enter a binary number: ");
            String binary = input.nextLine();
            System.out.println(binary + " is decimal " + binaryToDecimal(binary));
          }
    }
    public static int binaryToDecimal(String binaryString)
    {
        return binaryToDecimal(binaryString, 0, binaryString.length() - 1);
    }
    public static int binaryToDecimal(String binaryString, int low, int high)
    {...7 lines }
}
```

Output

Run (MP1-F22-1.0) | Debugger Console | recursion (run)

```
run:
Enter a binary number: 101
101 is decimal 5
Enter a binary number: 1111
1111 is decimal 15
Enter a binary number: 011
011 is decimal 3
Enter a binary number: 000111
000111 is decimal 7
Enter a binary number:
```

4. Modify the selection sort done in the lab and have it doing the sorting with 1 recursive call, not two. That is, one recursive call will have to be replaced by a loop.

5. Draw a diagram for the function calls of the selection sort with the 2 recursive calls in it as implemented in the lab. You have to show all variables at all function calls.
   The array to sort is {10,1,20,3}Upload a Word document.

6. ( 10 points) The given the Java-FX class Maze finds a path in a maze from square (0,0), to square (7,7) ( top-left to bottom right)

The maze is represented by an 8 * 8 board.
The path meets the following conditions:

- The path is between the upper-left corner cell and the lower-right corner cell in the maze.

- The program enables the user to place or remove a mark on a cell. A path consists of adjacent unmarked cells. Two cells are said to be adjacent if they are horizontal or vertical neighbors, but not if they are diagonal neighbors.

- The path does not contain cells that form a square.

**Add code to the given solution to**

1. Find a path from square (0,7), to square (7,0) ( top-right to bottom left). Add a proper button to find the path. The color of the path is green not red.

2. When entrance squares (0,0) or ( 0, 7) are blocked with X or exit squares ( 7, 0) or ( 7, 7) are blocked to display a message for the user to unblock the square and not proceed to find a path until the entrance and/or exit square is/are cleared.

3. If/when green and red path having intersecting squares, the squares of intersection should become yellow, not green or red.

Maze.html is posted.