Question 1 (10 marks)

a) Write a Java program called Dav.java which reads a positive integer, d, from the command line, and generates the 2^d **Davison** sequence using **recursion**.

Code:

Davison.java

```java
public class Davison {
    Run | Debug
    public static void main(String[] args) {
        int inpt = 5;
        String numString = "0";
        numString = Davison_Seq(numString, inpt);
        System.out.println(numString);
    }

    public static String Davison_Seq(String numString,int inpt){
        int i = 0;
        if(inpt == 0 )
            return numString;
        else{
            try{
                System.out.println(numString);
                char[] bitwise = numString.toCharArray();
                int str_length = numString.length();
                while(i < str_length){
                    if(bitwise[i] == '1')
                        bitwise[i] = '0';
                    else if (bitwise[i] == '0')
                        bitwise[i] = '1';
                    else
                        bitwise[i] = '@';
                    i++;
                }
                String newString = String.valueOf(bitwise);
                numString = numString.concat(newString);
            }
            catch (Exception e){
                System.out.println("Error in"+e);
            }
            return Davison_Seq(numString,inpt-1);
        }
    }
}
```

Output:

```
PS C:\Users\film2\Downloads\EXAlgo> c:; cd 'c:\Users\film2\Downloads\EXAlgo'; & 'c:\Users\film2\.vscode\extensions\vscjava.vscode-java-debug-0.31.0\scripts\launcher.bat' 'C:\Users\film2\AppData\Local\Programs\AdoptOpenJDK\jdk-11.0.8.10-hotspot\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:51410' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\film2\AppData\Roaming\Code\User\workspaceStorage\3fdbdd21886d2d49f984d0b00291d6e9\redhat.java\jdt_ws\EXAlgo_172f5aea\bin' 'Davison'
0
01
0110
01101001
0110100110010110
0110100110010110100101100110100 1
```

b) Calculate the **big-Oh** running time for the program in terms of the size of d.

- Term size of d,big-Oh running time คือ O(n 2)
- N loop of while (I < str_length),big-Oh running time คือ O(n)

  Running time = (term size of d*number of loop)

  = $O(n^2) * O(n)$

  = $O(n^3)$

Question 2 (10 marks)

You have a bag containing the numbers 1, 2, 3, ..., n, with each number appearing exactly once. However, one of the numbers has been removed from the bag.

Write **pseudocode** that finds which number is missing.The algorithm must have running time O(n).

pseudo.txt

```
step 1 : Initial a array of num
step 2 : Initial a value of check=0, sum=0, total=0, n=0 and i=0
step 3 : Assign length of num to num
step 4 : While loop if n lower than i
step 5 : sum += array of num
step 6 : check = (n+1)*(n+2)/2
step 7 : total = check - sum
step 8 : print total

```

Output: 6

Question 3

a) Implement the **explore()** function in MazeSearcher.java. It must use **backtracking** to search for the exit from the maze, and build a path as it searches. A path is a list of coordinates.

Code:

```
private boolean explore(Maze maze, int x, int y,ArrayList<Coord> path) {

  if(!maze.isValidLoc(x, y)||maze.isWall(x, y)||maze.wasVisited(x, y))
    return false;

  path.add(new Coord(x, y));
  maze.setVisited(x, y);
  if(maze.isExit(x, y))
    return true;

  for(int[] step:STEPS){
    Coord coord = getNextCoord(x,y,step);
    if(explore(maze,coord.getX(), coord.getY(), path))
      return true;
  }
  path.remove(path.size()-1);
  return false;
}  // end of explore()
```

Output:

Maze1.txt

```
PS C:\Users\film2\Downloads\EXAlgo\Maze\Maze> javac *.java
PS C:\Users\film2\Downloads\EXAlgo\Maze\Maze> java MazeSearcher maze1.txt
#S########
#.....   #
# ###.## #
# # .. # #
# # .# # #
# ##.#####
# # ...  #
# # #..# #
#####.####
#   #....E
# #    # #
##########
```

Maze2.tx.

```
PS C:\Users\film2\Downloads\EXAlgo\Maze\Maze> java MazeSearcher maze2.txt
#S##########################
#.#      #                 #
#.# ####   ############### #
#.# #    # #               #
#.# #### # # ###############
#.# #    # # #.............#
#.# # #### ###.###########.#
#.# # #........#          .#
#.......#################.#
#########    # # #       #.#
# #       #### # ####### #.#
# # ### ###  # # #.......#.#
#   #       ##...#.#####.#.#
##### #######..#.#..   #.#.#
#     #.......##.##.####.#.#
# #####.######## ....#   ...#
#       .#       ############
#######.#########        # #
#.......#........ ######## #
#.#######.######.##...#....E
#.........#     ....#...## #
############################
```

b) Explain in words (and perhaps with diagrams) why the Maze class includes the methods wasVisited() and setVisited(). *Hint*: if your explore() function does not use these functions, then your search may take a very, very long time.

- Maze class มีวิธีหารที่ต่อเนื่องในฟังก์ชันwasVisited()ทำไห้ไฟล์mazeจะจำเส้นทางที่เคยไปมาแล้ว จะทำให้ไม่ไปซ่ำกับเส้นทางเดิม และ ฟังค์ขันsetVisited()ทำไห้mazeบันทึกเส้นทางระหว่างการต้นหาเส้นทางอยู่ด้วยซึ่ง มี 2 วิธีคือการช่วยหาเส้นทางที่ถูกต้อง และ หาเส้นทางที่สั้นที่สุดที่จะทำให้ไปถึงจุดหมายได้อย่างรวดเร็ว