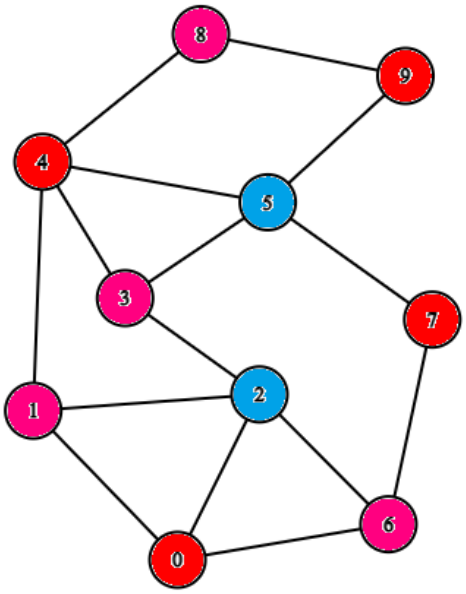
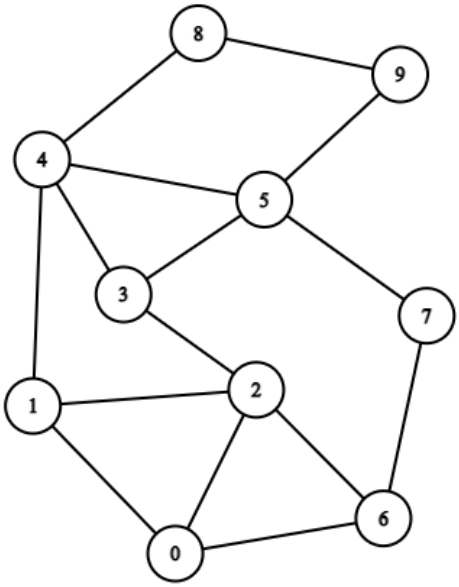
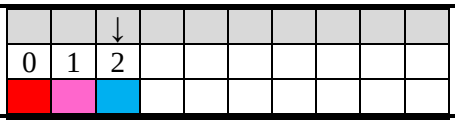
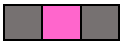
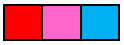
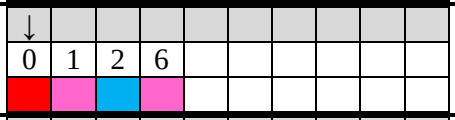
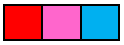
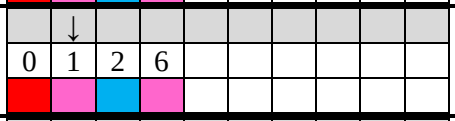
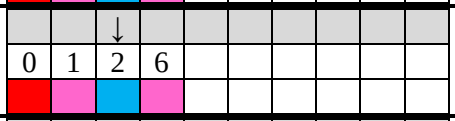
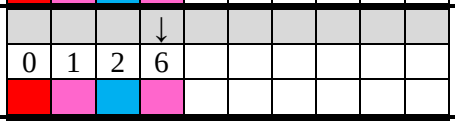
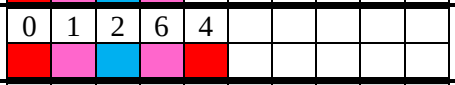
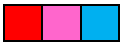
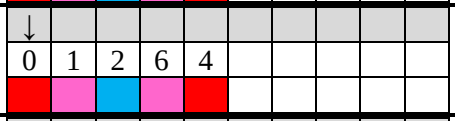
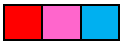
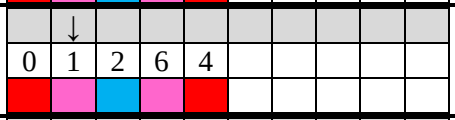
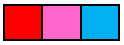
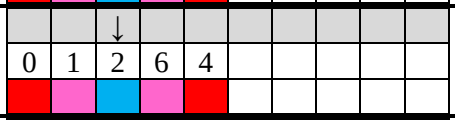
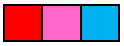
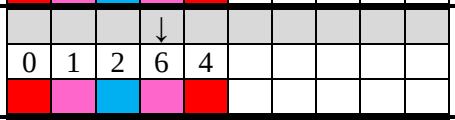
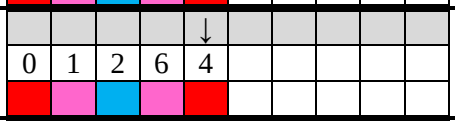
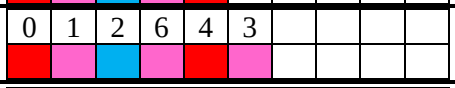
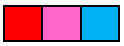
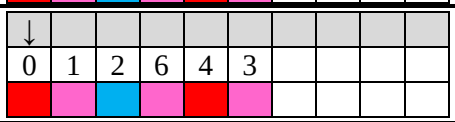
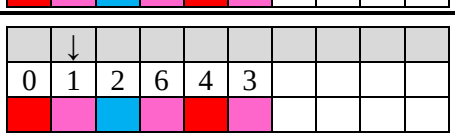
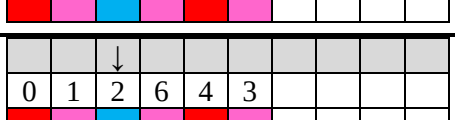
Laboratory 5 – Zsok Alina-Valentina

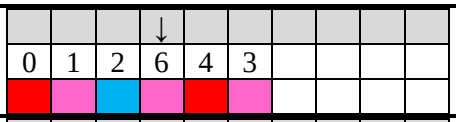
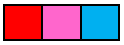
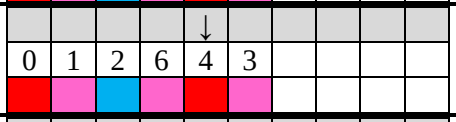
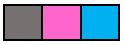
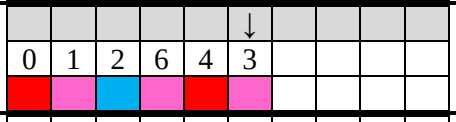
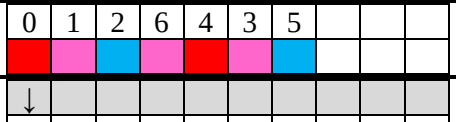
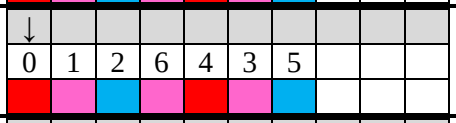
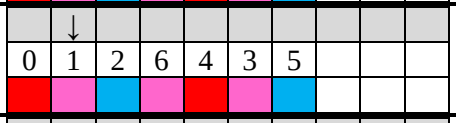
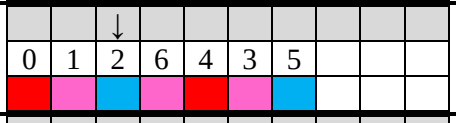
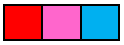
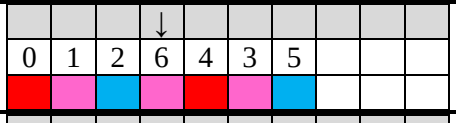
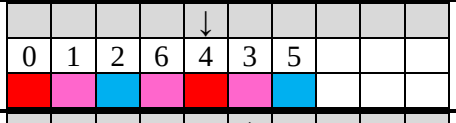
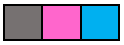
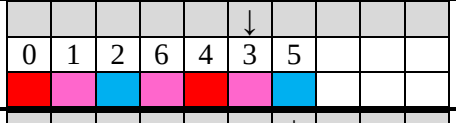
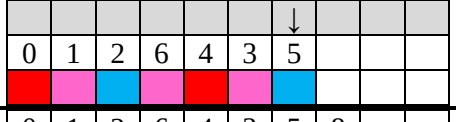
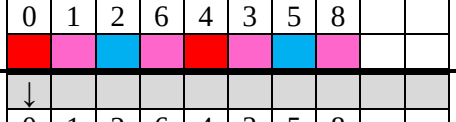
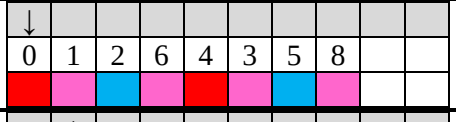
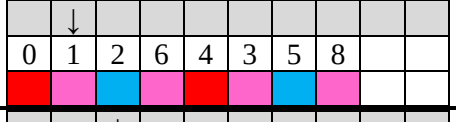
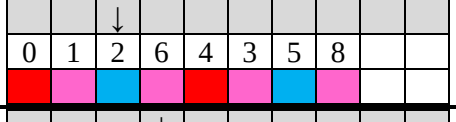
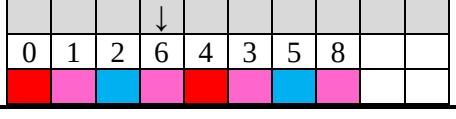


Problem 3 - Given an undirected graph, find a vertex coloring with minimum number of colors.



It.	current Vertex	previous	colored Vertex	edge (v1, v2)	used Colors
		0			
1	1	0			
1.1		↓ 0	0	True	
2	2	0 1			
2.1		↓ 0 1	0	True	
2.2		↓ 0 1	1	True	
3	6	0 1 2			
3.1		↓ 0 1 2	0	True	
3.2		↓ 0 1 2	1	False	

3.3			2	True	
4	4				
4.1			0	False	
4.2			1	True	
4.3			2	False	
4.4			6	False	
5	3				
5.1			0	False	
5.2			1	False	
5.3			2	False	
5.4			6	False	
5.5			4	True	
6	5				
6.1			0	False	
6.2			1	False	
6.3			2	False	

6.4			6	False	
6.5			4	True	
6.5			3	True	
7	8				
7.1			0	False	
7.2			1	False	
7.3			2	False	
7.4			6	False	
7.5			4	True	
7.6			3	False	
7.7			5	False	
8	9				
8.1			0	False	
8.2			1	False	
8.3			2	False	
8.4	9		6	False	

8.5		0 1 2 6 4 3 5 8	4	False
8.6		0 1 2 6 4 3 5 8	3	False
8.7		0 1 2 6 4 3 5 8	5	True
8.8		0 1 2 6 4 3 5 8	8	True
9	7	0 1 2 6 4 3 5 8 9		
9.1		0 1 2 6 4 3 5 8 9	0	False
9.2		0 1 2 6 4 3 5 8 9	1	False
9.3		0 1 2 6 4 3 5 8 9	2	False
9.4		0 1 2 6 4 3 5 8 9	6	True
9.5		0 1 2 6 4 3 5 8 9	4	False
9.6		0 1 2 6 4 3 5 8 9	3	False
9.7		0 1 2 6 4 3 5 8 9	5	True
9.8		0 1 2 6 4 3 5 8 9	8	False
9.9		0 1 2 6 4 3 5 8 9	9	False
		0 1 2 6 4 3 5 8 9 7		

List of colors used									
Re	Pi	Bl	Pu	Gr	Ye	Or	Br	Bk	Wh

- **Source Code Explanation**

```
public String colorGraphMinimum(){
    ArrayList<Color> colorsUsed = new ArrayList<>();        // a list of all the colors used overall
    ArrayList<ColoredVertex> previous = new ArrayList<>();  // a list of previous nodes that were colored
    ArrayList<ColoredVertex> allNodes = new ArrayList<>();  // a list of all the nodes from the undirected graph

    for(int vertex : this.vertices){                        // adds all the nodes from the list
        ColoredVertex newVertex = new ColoredVertex(vertex);
        allNodes.add(newVertex);
    }

    allNodes.get(0).setColor(this.colors.get(0));
    previous.add(allNodes.get(0));
    allNodes.remove( index: 0);
    colorsUsed.add(this.colors.get(0));
```

Step 1

```
    for(ColoredVertex uncoloredVertex : allNodes){
        ArrayList<Color> copyUsedColors = new ArrayList<>(colorsUsed);  // copy of used color list
        for(ColoredVertex coloredVertex : previous){
            if(this.existsEdgeBetweenTwoVertices(uncoloredVertex.vertex,coloredVertex.vertex)){
                if(copyUsedColors.contains(coloredVertex.color)){
                    copyUsedColors.remove(coloredVertex.color);
                }
            }
        }
```

Step 2

```
        if(copyUsedColors.size() != 0){
            uncoloredVertex.setColor(copyUsedColors.get(0));
        }else{
            uncoloredVertex.setColor(this.colors.get(colorsUsed.size()));
            colorsUsed.add(this.colors.get(colorsUsed.size()));
        }
        previous.add(uncoloredVertex);
    }
```

Step 3

- **Step 1 – Initializing all the lists**

a) sets the *color of the first node* from the undirected graph with the *first available color*;
b) adds the *first node* to the *previous list*;
c) adds the *used color* to the *colorsUsed list*;

- **Step 2 – Checking if the current chosen vertex is a neighbor of a previous nodes**

a) selects the *next uncolored vertex* from the list;
b) makes a copy of the colorsUsed list (*copyUsedColors*);
c) checks if exists an edge between the *uncolored vertex* and the *nodes* from the *previous list*;
   i. if an edge exists then it removes the color used for the other vertex
   ii. otherwise, it goes to the next node from the previous list

- **Step 3 – Colors the uncolored vertex**

a) checks if any (already) *used colors* is available for usage;
   i. if there is an (already) used color available then it uses the *first one from the list*;
   ii. otherwise, the uncolored vertex is colored with the next unused color from the colors list and add the color to the colorsUsed list;

b) adds the now colored vertex and adds it to the previous list;