Name and, if possible, ID#:

Problem 2

Colors in Java can be represented by objects of type ***Color***. Each such object contains the ***red***, ***green*** and ***blue*** components of the corresponding color as integer values from **0** to **255**. Consider below a Java code that creates and initializes a rectangular array of ***Color*** type:

```
import java.util.Scanner;
import java.awt.Color;

public class Colors {

public static void main(String args[]) {
      Scanner in = new Scanner(System.in);

// Read number of rows and columns and create a Color array of such size
      Color[][] c = new Color[in.nextInt()][in.nextInt()];

// For each element read the red, green and blue components as integers and
// create a Color object by calling Color(int, int, int) constructor
      for (int row = 0; row < c.length; row++)
            for (int col = 0; col < c[0].length; col++)
                  c[row][col] = new Color(in.nextInt(), in.nextInt(), in.nextInt());
// TO BE CONTINUED
}
}
```

Continue with a Java code that creates another array ***Color[][] g*** of the same size and fills it with gray equivalents of the colors from the array ***Color[][] c***. To get a grey equivalent of a given color ***c[i][j]***, it is enough to construct a ***Color*** object, whose red, green and blue components all are equal to the calculated average of red, green and blue components of the initial ***c[i][j]***. Use ***int getRed()***, ***int getGreen()*** and ***int getBlue()*** methods of class ***Color***.

```
int average;
Color[][] g = new Color[c.length][c[0].length];
Int average;
for (int row = 0; row < c.length; row++)
    for (int col = 0; col < c[0].length; col++)
    { average = (c[row][col].getRed() + c[row][col].getGreen() +
                 c[row][col].getBlue()) / 3;
      g[row][col] = new Color(average, average, average); }
```

5

OOP. MT1. 1705.15. H088

Name and, if possible, ID#:

Problem 3

Similar to files, strings also can be related to streams in C++, this time using ***stringstream*** objects. Particularly, it is enough to create an object of type ***istringstream*** to organize formatted reading from a string. Consider, for example, a C++ code below:

```
#include <string>
#include <sstream>
#include <iostream>
using namespace std;

void main()
{
      string text = "Before_increment: 199999999", word;
      int num;
      istringstream tokens(text);

      tokens >> word >> num;
      cout << "After " << word.substr(7) << num + 1 << endl;
}
// After increment:200000000
```

Write a C++ function ***double value(string expression)*** that takes as its argument a string representing an arithmetic expression, evaluates it and returns its value. The expression includes only '+' and '–' operations and double operands, both positive and negative. The operands and operations are delimited by spaces.

For example, ***value("5.1 –(-0.7)+ 1.2")*** results in ***7.0***.

6

```
double value (string expression)
{ double num;
  istringstream tokens (expression);
  double result = 0.0;
  char sign = '';
  while (tokens >> num >> sign && tokens >> num )
  { if (sign == '+')
      result = result + num;
    else if (sign == '-')
      result = result - num; }
  return (result);
}
```

OOP.MT1.170215.H088