პროგრამირების აბსტრაქციები
სემინარის ამოცანები #5

1 Minesweeper-ის დაფის შედგენა

ა) დაწერეთ ფუნქცია რომელიც მოცემული სიგრძის და სიგანის Minesweeper-ის დაფას შეავსებს შემთხვევით განლაგებული ბომბებით. დაფის თითოეულ უჯრაზე ბომბის ყოფნის ალბათობა არგუმენტად უნდა გადაეცემოდეს ფუნქციას.

```
#include "grid.h"
#include "random.h"

void createMineFieldWithProbability(Grid<bool>& result, double probability) {
    for (int i = 0; i < result.numRows(); i++) {
        for (int j = 0; j < result.numCols(); j++) {
            result[i][j] = randomChance(probability);
        }
    }
}
```

ბ) დაწერეთ წინას მსგავსი ფუნქცია, ოღონდ ამჯერად ბომბის ალბათობის მაგივრად ბომბების სრული რაოდენობა გადაეცით ფუნქციას. შეგიძლიათ ჩათვალოთ რომ ბომბების რაოდენობა დაფაზე არსებულ ადგილებზე ბევრად ნაკლებია და ამიტომ ცარიელი ადგილის ძებნაზე ბევრი დრო არ დაგეხარჯებათ.

```
void createMineFieldWithCount(Grid<bool>& mines, int numMines) {
    while (numMines > 0) {
        int y = randomInteger(0, mines.numRows() - 1);
        int x = randomInteger(0, mines.numCols() - 1);

        if (!mines[y][x]) {
            numMines--;
            mines[y][x] = true;
        }
    }
}
```

2 Minesweeper-ის დაფის ცარიელი ადგილების რიცხვებით შევსება

დაწერეთ ფუნქცია, რომელსაც გადაეცემა წინა ამოცანაში შექმნილი Minesweeper-ის დაფა (grid) და ფუნქცია შეავსებს მეორე (იგივე ზომის) დაფას მთელი რიცხვებით იმის მიხედვით თუ მოცემული უჯრის გარშემო 3x3 კვადრატში რამდენი ბომბია. თუ მოცემულ უჯრაზე ბომბია ამ უჯრაში -1 უნდა ჩაიწეროს.

```
int getSingleHint(Grid<bool>& mines, int x, int y) {
    if (mines[y][x])
        return -1;

    int result = 0;
    for (int i = x - 1; i <= x + 1; i++) {
        for (int j = y - 1; y <= y + 1; j++) {
            if (i >= 0 && i < mines.numCols()
                && j >= 0 && j < mines.numRows()) {
                if (mines[j][i])
                    result++;
            }
        }
    }

    return result;
}

void calculateHints(Grid<int>& hints, Grid<bool>& mines) {
    for (int y = 0; y < mines.numRows(); y++) {
        for (int x = 0; x < mines.numCols(); x++) {
            hints[y][x] = getSingleHint(mines, x, y);
        }
    }
}
```

3 ბომბების რაოდენობა ნებისმიერ მართკუთხედში

ა) დაწერეთ ფუნქცია რომელიც, Minesweeper-ის დაფის მიხედვით, ნებისმიერი უჯრისთვის, გამოთვლის ზედა მარცხენა უჯრის და ამ უჯრის მომცველ უმცირეს მართკუთხედში რამდენი ბომბი არის. ეს რიცხვები ცალკე დაფაში შეინახეთ.

```
void getTopLeftRectangleSums(Grid<int>& rectangleSums, Grid<bool>& mines) {
    for (int x = 0; x < mines.numCols(); x++) {
        for (int y = 0; y < mines.numRows(); y++) {
            if (x == 0 && y == 0) {
                rectangleSums[y][x] = mines[y][x];
            } else if (y == 0) {
                rectangleSums[y][x] = rectangleSums[y][x-1]
                    + mines[y][x];
            } else if (x == 0) {
                rectangleSums[y][x] = rectangleSums[y-1][x]
                    + mines[y][x];
            } else {
                rectangleSums[y][x] = rectangleSums[y-1][x]
                    + rectangleSums[y][x-1] - rectangleSums[y-1][x-1]
                    + mines[y][x];
            }
        }
    }
}
```

ბ) წინა ნაწილში აგებული დაფის გამოყენებით დაწერეთ ფუნქცია, რომელიც ნებისმიერი მართკუთხედისთვის (რომლის ზედა მარცხენა და ქვედა მარჯვენა წერტილებს არგუმენტებად გადავცემთ ფუნქციას) გამოთვლის ამ მართკუთხედში რამდენი ბომბი არის.

```
int getRectangleSum(Grid<int>& rectangleSums, int x, int y) {
    if (x < 0 || y < 0) {
        return 0;
    } else {
        return rectangleSums[y][x];
    }
}

void getBombsInRecangle(Grid<int>& rectangleSums,
                        int x1, int y1, int x2, int y2) {
    return getRectangleSum(rectangleSums, x2, y2)
            - getRectangleSum(rectangleSums, x1-1, y2)
            - getRectangleSum(rectangleSums, x2, y1-1)
            + getRectangleSum(rectangleSums, x1-1, y1-1);
}
```