Description

sample code of main.cpp

```
#include <iostream>

#include "function.h"


int main()

{

    Block b{4, "@OO@" "OOOO" "OOOO" "@OO@"}; // create a 4x4 pattern, which is rotati
on invariant

    /*std::cout << b << std::endl;*/


    if (invariant(b)) // checking if two patterns are equivalent under rotation

        std::cout << "INVARIANT" << std::endl;

    else

        std::cout << "VARIANT" << std::endl;


    Block c{4, "@XX@" "OOOO" "OOOO" "@OO@"}; // create a 4x4 pattern, which is rotati
on variant

    /*std::cout << c << std::endl;*/


    if (invariant(c)) // checking if two patterns are equivalent under rotation

        std::cout << "INVARIANT" << std::endl;

    else

        std::cout << "VARIANT" << std::endl;


}
```

code of function.h

```
#ifndef _BLOCKCLASS_
#define _BLOCKCLASS_
#include <iostream>
#include <algorithm>
#include <utility>
class Block {
private:
    int size;
    char** pattern; // array of pointers to buf
    char* buf;      // 2D pattern stored in 1D raster scan order
public:
    Block(): size{0}, pattern{nullptr}, buf{nullptr} {}

    Block(int sz, const char* pat):
        size{sz}, pattern{new char* [size]}, buf{new char[size*size]}
    {
        // std::cout << "custom constructor\n";
        for (int i=0; i<size*size; i++) buf[i]=pat[i];
        for (int i=0; i<size; i++) {
            pattern[i] = (char*) &buf[i*size];
        }
    }

    Block(const Block &b):
        size{b.size}, pattern{new char* [size]}, buf{new char[size*size]}
    {
        // std::cout << "copy constructor\n";
        for (int i=0; i<size*size; i++) buf[i]=b.buf[i];
```

```
        for (int i=0; i<size; i++) {
            pattern[i] = (char*) &buf[i*size];
        }
    }

    Block& operator=(Block& c) {
        Block b{c};
        // std::cout << "copy assignment\n";
        std::swap(buf, b.buf);
        std::swap(pattern, b.pattern);
        size = b.size;
        return *this;
    }

    // rvalue reference
    Block(Block&& b): size{b.size}, pattern{b.pattern}, buf{b.buf}
    {
        // std::cout << "move constructor\n";
        b.size = 0;
        b.pattern = nullptr;
        b.buf = nullptr;
    }

    // rvalue reference
    Block& operator=(Block&& b) {
        // std::cout << "move assignment\n";
        if (this != &b) {
            delete [] buf;
```

```
        delete [] pattern;


        buf = b.buf;
        pattern = b.pattern;
        size = b.size;


        b.buf = nullptr;
        b.pattern = nullptr;
        b.size = 0;
    }
    return *this;
}


~Block()
{
    // std::cout << "destructor\n";
    delete [] buf;
    delete [] pattern;
}


friend std::ostream& operator<<(std::ostream& os, Block& b) {
    for (int i=0; i<b.size; i++) {
        for (int j=0; j<b.size; j++) {
            os << b.pattern[i][j];
        }
        os << std::endl;
    }
    return os;
```

```
    }


    // the task is to implement the following two functions

    void clockwise90();

    friend bool invariant(const Block& a);


};

#endif
```

Input

Output

Sample Input

EOF

Sample Output

EOF