第二次作业

第一题

```cpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;
class ChessBoard {
public:
    int** map;
    ChessBoard(int s) :cb_size(s),map(new int*[s]) {
        for (int i = 0; i < cb_size; i++) {
            map[i] = new int[cb_size];
        }
        for (int i = 0; i < cb_size; i++) {
            for (int j = 0; j < cb_size; j++) {
                map[i][j] = 0;
            }
        }
    }

    ~ChessBoard() {}
    void show();
    int get_size() { return cb_size-5; }
private:
    int cb_size;
};

class playerU {
public:
    typedef struct Pos {
```

```
        int x;
        int y;
    };
    playerU(string name,int type) :_name(name),_type(type) {}
    ~playerU() {}
    string& get_name() { return _name; }
    int get_type() { return _type; }
    int setchess(int x, int y, ChessBoard& cb);//0  -> continue ; 1  -> win ; 2  ->  re
    int judge(ChessBoard& cb);//0  -> continue ; 1  -> win
    bool isValid(int x,int y,ChessBoard& cb);
private:
    string _name;
    int _type;
    vector<Pos> pieces;
};

void ChessBoard::show() {
    for (int i = 0; i < get_size(); i++) {
        for (int j = 0; j < get_size(); j++) {
            switch (map[i][j]) {
            case 1:cout << "* "; break;
            case 2:cout << "# "; break;
            case 0:cout << "+ ";
            }
        }
        cout << endl;
    }
}
int playerU::setchess(int x, int y, ChessBoard& cb) {
    if (isValid(x, y,cb)) {
        cb.map[x][y] = _type;
        pieces.push_back({ x,y });
    }
    else {
        return 2;
    }

    return judge(cb);
}
int playerU::judge(ChessBoard& cb) {
    for (int i = 0; i < cb.get_size(); i++) {
        for (int j = 0; j < cb.get_size(); j++) {
```

```
            if ((_type == cb.map[i][j] && _type == cb.map[i + 1][j] && _type == cb.map[i + 2][j] && _type == cb.map[i + 3][j] && _type == cb.map[i + 4][j]) ||
                (_type == cb.map[i][j] && _type == cb.map[i][j + 1] && _type == cb.map[i][j + 2] && _type == cb.map[i][j + 3] && _type == cb.map[i][j + 4]) ||
                (_type == cb.map[i][j] && _type == cb.map[i + 1][j + 1] && _type == cb.map[i + 2][j + 2] && _type == cb.map[i + 3][j + 3] && _type == cb.map[i + 4][j + 4])
                )return 1;
        }
    }
    return 0;
}
bool playerU::isValid(int x, int y,ChessBoard& cb) {
    return (x <= cb.get_size()-1 && y <= cb.get_size()-1&&!cb.map[x][y]) ? true : false;
}
int main()
{
    int size;
    cout << "input the size of the board"<<endl;
    cin >> size;
    ChessBoard cb(size+5);
    playerU p1("alex", 1);
    playerU p2("bob", 2);
    int x, y;
    while (1) {
        re1:
        cout << "player 1:" << endl;
        cin >> x >> y;
        switch (p1.setchess(x, y, cb)) {
        case 1:
            cout << "Player1 win"<<endl;
            goto end;
        case 2:
            cout << "illegal position,please retry"<<endl;
            goto re1;
        case 0:
            cb.show();
            break;
        }

```

```
        re2:
        cout << "player 2:" << endl;
        cin >> x >> y;
        switch (p2.setchess(x, y, cb)) {
        case 1:
            cout << "Player2 win"<<endl;
            goto end;
        case 2:
            cout << "illegal position,please retry" << endl;
            goto re2;
        case 0:
            cb.show();
            break;
        }
    }
    end:
    cb.show();
}


```

第二题

书上代码有误，应为

Date::Date(*const* Date& D)

(1)

```
Constructor Called. Address=0x000000311A12F988
Copy Constructor Called. Address=0x000000311A12FA78
Copy Constructor Called. Address=0x000000311A12FAC4
Destructor Called. Address=0x000000311A12FA78
Destructor Called. Address=0x000000311A12FAC4
Destructor Called. Address=0x000000311A12F988
```

第一处：创建today

第二处：形参复制

第三处：Date(A)复制

第四处：形参析构

第五处：复制的Date(A)析构

第六处：today析构

(2)

(1)

```
Constructor Called. Address=0x000000B9724FF708
Copy Constructor Called. Address=0x000000B9724FF7F8
Copy Constructor Called. Address=0x000000B9724FF5B8
Copy Constructor Called. Address=0x000000B9724FF844
Destructor Called. Address=0x000000B9724FF5B8
Destructor Called. Address=0x000000B9724FF7F8
Destructor Called. Address=0x000000B9724FF844
Destructor Called. Address=0x000000B9724FF708
```

(2)

```
Constructor Called. Address=0x0000008C8FCFF8D8
Copy Constructor Called. Address=0x0000008C8FCFF9C8
Copy Constructor Called. Address=0x0000008C8FCFF788
Destructor Called. Address=0x0000008C8FCFF788
Destructor Called. Address=0x0000008C8FCFF9C8
Destructor Called. Address=0x0000008C8FCFF8D8
```

形参与B析构顺序相反

(3)

```
Constructor Called. Address=0x0000008C8FCFF8D8
Copy Constructor Called. Address=0x0000008C8FCFF9C8
Copy Constructor Called. Address=0x0000008C8FCFF788
Destructor Called. Address=0x0000008C8FCFF788
Destructor Called. Address=0x0000008C8FCFF9C8
Destructor Called. Address=0x0000008C8FCFF8D8
```