計算機程式語言

作　　業　　五

姓名	潘廣霖
學號	B03203004
原始程式檔名	HW05_002.cpp (_ = [A-D])

評　分　項　目

分數比重	項　　目	得　分
40%	程式是否能正確執行?	
40%	程式之使用者介面與輸出結果?	
	是否有繳交原始程式檔與執行檔?	
	程式中的註解是否恰當?	
	程式之結構與邏輯是否正確?	
	程式碼的格式是否合乎要求?	
20%	程式之綜合評分	
總　分		

評語:

A.

```
//=================================================================
//  PROGRAMMER  : 潘廣霖
//  DATE        : 2015-11-27
//  FILENAME    : HW05A002.CPP
//  DESCRIPTION : This is an interactive Game of Life simulator!
//=================================================================

#include "stdafx.h"

#include<iostream>
#include<string>
using namespace std;
using namespace System;

static const int SIZE = 20;
static const int SIZE_PAD = SIZE + 2;

void outputStr(String ^str, bool setCur = false) {
    if (setCur) Console::SetCursorPosition(0, 1);
    Console::Write(str);
}

void writeLine(int n, String ^ s) {
    Console::SetCursorPosition(0, n);
    outputStr(s);
}

void setGenStr(int n) {
    outputStr(L"Generation: ", true);
    Console::Write(n);
}

void getPos(int& posLeft, int& posTop) {
    posLeft = Console::CursorLeft;
    posTop = Console::CursorTop;
}

inline void setPos(int& posLeft, int& posTop) {
    Console::SetCursorPosition(posLeft, posTop);
}

void clearLineUntilEnd(int l, int t) {
    for (int i = l; i < Console::BufferWidth; i++) {
        cout << ' ';
    }
    setPos(l, t);
}

void updateCell(int x, int y, int newState, int lnOffset) {
    Console::SetCursorPosition(y * 2, lnOffset + x + 1);
    cout << (newState ? "¡½" : "¡¼");
}

void simulateCells(int cell[][SIZE_PAD], int delta[][SIZE_PAD], int posTop) {
    for (int i = 1; i <= SIZE; i++) {
        for (int j = 1; j <= SIZE; j++) {
            delta[i][j] = 0;
            for (int di = -1; di <= 1; di++) {
                for (int dj = -1; dj <= 1; dj++) {
                    if (di == 0 && dj == 0) continue;
                    delta[i][j] += cell[i + di][j + dj];
                }

            }
```

```
            }
        }
        for (int i = 1; i <= SIZE; i++) {
            for (int j = 1; j <= SIZE; j++) {
                // determine the next state of the cell
                if (cell[i][j] && (delta[i][j] < 2 || delta[i][j] > 3)) {
                    cell[i][j] = 0;
                    updateCell(i - 1, j - 1, 0, posTop);
                }
                else if (!cell[i][j] && delta[i][j] == 3) {
                    cell[i][j] = 1;
                    updateCell(i - 1, j - 1, 1, posTop);
                }
            }
        }
    }

    void outputCells(int t[SIZE_PAD][SIZE_PAD]) {
        Console::SetCursorPosition(0, 2);
        for (int i = 1; i <= SIZE; i++) {
            for (int j = 1; j <= SIZE; j++) {
                cout << (t[i][j] ? "¡½" : "¡¼");
            }
            cout << endl;
        }
    }

    int main(array<String ^> ^args) {
        int cnt = 0;
        int cell[SIZE_PAD][SIZE_PAD] = { 0 };
        int delta[SIZE_PAD][SIZE_PAD];
        int x = -1, y = -1;
        int posLeft;
        int posTop;

        writeLine(0, "Game of Life");
        outputStr(L"Input (x,y) to toggle the cell, input (-1,-1) to start simulation:
", true);
        getPos(posLeft, posTop);
        outputCells(cell);
        setPos(posLeft, posTop);

        // input state
        while (cin >> x >> y
            && x >= 0 && y >= 0
            && x <= SIZE && y <= SIZE) {
            cell[x+1][y+1] = !cell[x+1][y+1];
            // update only the cell; assuming wide characters
            updateCell(x, y, cell[x+1][y+1], posTop);
            setPos(posLeft, posTop);
            // clear previous output, allowing next input
            clearLineUntilEnd(posLeft, posTop);
        }

        cin.ignore(1);

        // simulation state
        string s;
        while (true) {
            outputStr(L"===Emulation=== Go to next n generations (Ctrl+Z to end) [1]:
", true);
            getPos(posLeft, posTop);
            clearLineUntilEnd(posLeft, posTop);

            bool stopSig = !getline(cin, s);
            int gen = 0;
            if (!s.size()) gen = 1;
            for (int i = 0, n = s.size(); i < n; i++) {
```

```
            if (s[i] >= '0' && s[i] <= '9') gen = gen * 10 + (s[i]-'0');
            else {
                stopSig = true;
                break;
            }
        }

        if (stopSig) break;

        while (gen--)
            simulateCells(cell, delta, posTop);
    }

    return 0;
}
```

B.

```
//============================================================
//  PROGRAMMER  : 潘廣霖
//  DATE        : 2015-11-27
//  FILENAME    : HW05B002.CPP
//  DESCRIPTION : Compute a magic square of odd cells.
//============================================================

#include "stdafx.h"

#include<iostream>
#include<iomanip>
#include<cstdlib>

using namespace std;

#ifdef _MSC_VER
int _tmain(int argc, _TCHAR* argv[]) {
#else
int main() {
#endif // _MSC_VER
    int** magic;

    int n, n2;
    cout << "Input the edge N = ";
    cin >> n;
    n2 = n * n;

    if (!(n%2)) {
        cout << "n must be odd! exiting..." << endl;
        return 1;
    }

    // allocating an n x n matrix dynamically
    magic = new int* [n];
    for (int i = 0; i < n; i++) {
        magic[i] = new int[n];
        for (int j = 0; j < n; j++)
            magic[i][j] = 0;
    }

    // filling number
    int x = 0, y = n/2;
    for (int i = 0; i < n2; i++) {
        magic[x][y] = i+1;
        x = (x+n-1) % n;
        y = (y+1) % n;
        if (magic[x][y] > 0) {
            x = (x+2) % n;
            y = (y+n-1) % n;
        }
    }

    // printing
    cout << "Here is " << n << " x " << n << " magic square!" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            cout << setw(5) << right << magic[i][j];
        cout << endl;
    }

    system("pause");
}
```