系級：電機2乙 姓名：蔡慶霖 學號：F84051053

Project2-Sudoku

**1.操作手冊**

Compilation:

make

Execution:

./project2

執行後，輸入a(**隨機出唯一解數獨題**)或b(**從file讀入題目**)

若選擇從檔案讀入題目，須輸入檔案名稱

若選擇隨機出題，須**輸入挖洞數**或是輸入r(**挖洞數由程式隨機產生**)

題目產生之後印到螢幕與**problem.txt**

題目會再詢問**要開始玩(y)**或是**不玩直接看答案(n)**(答案印到螢幕與**answer.txt**)

遊戲進行時，有四種功能，**輸入(i)、刪除(d)、看答案(s)、離開程式(e)**。

e.g.

若欲在第2列、第1行輸入6，則輸入**i 2 1 6**

若欲刪除第2列、第1行的數，則輸入**d 2 1**

若欲看答案直接輸入s(答案印到螢幕與**answer.txt**)

若欲離開程式直接輸入e(什麼都不做**離開**)

當完成數獨，則印出Complete Sudoku!

**2.程式撰寫流程**

Step.1 讀取file測資，並判定是否為有效的題目(每行、每列、每宮格不可重複)

Step.2 解數獨測資(**使用recursive+候選數法+列、行、宮格屏除法**)

Step.3 利用recursive可將所有可能掃過，故可判定有多少解

Step.4 產生40~50holes的唯一解數獨題目(利用**挖洞法**)

Step.5 互動式(**可填、可刪除、可直接看答案、可選擇題目挖洞數**)

**3.作業心得**

本次作業著重在程式效率的部分，起初不打算用recursive的方法去寫的，畢竟recursive可能占用大量的記憶體，但後來實在想不到其他更合適，或是更簡便的寫法，所以最後依然選擇使用recursive。

然而，為了提升效率，我主要採用數獨解法中常見的候選數法，把每個空格當中可以填入的數字存到一個三維vector，並從最少可能的空格開始解，但這樣子的演算法，到了更多空格(45以上)時，仍然不夠有效率，因此再添加了屏除法原則，候選數可以再剔除一大批，少走了很多條錯誤的路。

說實在，這次做作業是第一次認真思考程式效率的問題，所以也花了不少時間去修改code，畢竟出題選用挖洞法，等於說每挖一個洞，就要檢查一次是不是唯一解(解一次)，若是挖20個洞就要解20次呢！幸好50個洞的題目依然可以一下子就出好，電腦的效率也挺驚人的，這次project2算是很大的磨練。

**4.Source Code說明**

a.數獨求解Recursive作法(求解與判斷解的數量)：

先做好前置作業，掃一次vector得出可能可以填入的數字存入pb，之後recursive內的find\_candidate便可直接取用，(solve function)遞迴每一層的PB(可填數)須採用call by value，因每層的可填數不可互相影響，而每次成功填一個數要重新修改候選數(candidate vector)才傳入下一層，call下一層function後須將原候選數與原填入數字回復。而recursive終止條件為當(Recursive層數-1 == 起初的空格數)(即層數>holeNum)或是無候選數直接return，最後可得出所有的result(為提高程式效率只要result>=2一律直接return)。

b.產生唯一解作法：

本程式從測資以挖洞法為主、並以鏡射法、旋轉法、行列對調、數字對調為輔產生數獨，測資本身有30holes，每挖一次洞或是每變換一次就解一次數獨判斷是否唯一解，若非唯一解則改挖其他洞，若是唯一解則繼續挖，直到挖洞數符合條件為止。

c.各function功能略說：

**//從pb提取候選數字，再利用屏除法加以篩選候選數**

**void find\_candidate(int row,int col,vector<int>& v,vector< vector<vector<int> > >& PB);**

**int inVec(int num,vector<int> v); //判斷某元素存在vector與否**

**void getholeNum(); //獲取空格數(可填位置)**

**void getpb();//獲取題目每個位置可填的數字**

**void getpbNum(); //獲取每個位置可填數字的數量**

**void getlist(); //獲取解空格的順序**

**void solve(int cur,vector<vector<int> > v,int hole,vector< vector<vector<int> > > PB); //遞迴求解**

**void getResult(); //前置作業+遞迴求解**

**void print\_question(); //印出數獨問題**

**void print\_result(); //印出數獨解答**

**//clear function於某空格填入數字後，剔除候選數**

**void clearRow(int value,int row,vector< vector<vector<int> > >& PB);**

**void clearCol(int value,int col,vector< vector<vector<int> > >& PB);**

**void clearBlock(int value,int row,int col,vector< vector<vector<int> > >& PB);**

**//check function 檢查sudoku每行列宮格有無數字重複**

**int checkCol(vector<vector<int> >& v);**

**int checkRow(vector<vector<int> >& v);**

**int checkBlock(vector<vector<int> >& v);**

**inline int checkSudoku(vector<vector<int> >& v);**

**void mirror(vector<vector<int> >& vec);//鏡射**

**void swap3(vector<vector<int> > &v); //對調三行(列)**

**void swap1(vector<vector<int> > &v); //對調一行(列)**

**void turnCol(vector<vector<int> >&v,int a,int b);//行隊調**

**void turnRow(vector<vector<int> >&v,int a,int b);//列對調**

**void changeNum(); //數字替換**

**void clockwise(vector<vector<int> >&v); //順時針旋轉90°**

**void dighole(vector<vector<int> >&v); //挖一個洞**