```
/**

    Sudoku Solver
        Abhishek Srikanth
        Class 12 - A
        Global Indian Int'l School

**/

#include <iostream>
#include <fstream>
#include <conio.h>

using namespace std;

class box
{
    public:
    int val;
    int nposib;
    int posib[9];
    box()   // sets val to ZERO, posib from 1-9
    {
        val = 0;
        nposib = 9;
        for(int i = 0; i < 9; ++i)
        {
            posib[i] = i + 1;
        }
    }
};
box sudoku[9][9];
box save[9][9];


void set_posib()
{
    // "Will now set posibilities for each value";

    for(int i = 0; i < 9; ++i)
    {
        for(int j = 0; j < 9; ++j)
        {
            if(sudoku[i][j].val!=0)     // IF SUDOKU[i][j] HAS A REAL VALUE
            {

                // Loop removes SUDOKU[i][j]'s value as a posib from corresponding row and column
                for(int m = 0; m < 9; ++m)
                {

                       if(sudoku[i][m].posib[ sudoku[i][j].val - 1 ] != 0)
                        {
                            sudoku[i][m].posib[ sudoku[i][j].val - 1 ] = 0;
                            sudoku[i][m].nposib--;      // change no. of possibilities
                        }
                        if(sudoku[m][j].posib[ sudoku[i][j].val - 1 ] != 0)
                        {
                            sudoku[m][j].posib[ sudoku[i][j].val - 1 ] = 0;
                            sudoku[m][j].nposib--;
                        }
                }

                /* Below body to find center point of corresponding quadrant */
                int Ci=-1,Cj=-1;
                if(i+1 == 1 || i+1 == 4 || i+1 == 7)
                    Ci = i+1;
```

```
                else if(i==1 || i==4 || i==7)
                    Ci = i;
                else if(i-1 == 1 || i-1 == 4 || i-1 == 7)
                    Ci = i-1;

                if(j+1 == 1 || j+1 == 4 || j+1 == 7)
                    Cj = j+1;
                else if(j==1 || j==4 || j==7)
                    Cj = j;
                else if(j-1 == 1 || j-1 == 4 || j-1 == 7)
                    Cj = j-1;
                /* Center point of quardrant is denoted by 'Ci' and 'Cj'    */

                for(int m = Ci-1; m < Ci+2; ++m)
                {
                    for(int n = Cj-1; n < Cj+2; ++n)
                    {
                        if(sudoku[m][n].posib[ sudoku[i][j].val -1 ] != 0)
                            {
                                sudoku[m][n].posib[ sudoku[i][j].val -1 ] = 0;
                                sudoku[m][n].nposib--;
                            }
                    }
                }
                /* Above nested loop accesses all values present in quadrant */

                sudoku[i][j].nposib = 0;
                for(int r = 0; r < 9; ++r)
                {
                    sudoku[i][j].posib[r] = 0;
                }
            }

        }
    }
}

void singletons()
{
    start_cuz_values_have_changed:
    for(int i = 0; i < 9; ++i)
    {
        for(int j = 0; j < 9; ++j)
        {
            if(sudoku[i][j].nposib == 1) // if only 1 possibility is present
            {
                for(int k = 0; k < 9; ++k)  // scan through posibilities
                {
                    if(sudoku[i][j].posib[k]!=0)  // If 'k'th possibility is NONZERO
                    {
                        sudoku[i][j].val = sudoku[i][j].posib[k];   // set value to that possibility
                        sudoku[i][j].nposib=0;                      // set number of possibilities to ZERO
                        for(int r = 0; r < 9; ++r)
                            sudoku[i][j].posib[r] = 0;

                        break;                                      // exit scanning possibilities
                    }
                }

            // To eliminate that value from corresponding ROW, COL, QUADRANT :

                    // Loop removes SUDOKU[i][j]'s value as a posib from corresponding row and column
                    for(int m = 0; m < 9; ++m)
                    {
                       if(sudoku[i][m].posib[ sudoku[i][j].val - 1 ] != 0)
                        {
```

```
                            sudoku[i][m].posib[ sudoku[i][j].val - 1 ] = 0;
                            sudoku[i][m].nposib--;
                        }
                        if(sudoku[m][j].posib[ sudoku[i][j].val - 1 ] != 0)
                        {
                            sudoku[m][j].posib[ sudoku[i][j].val - 1 ] = 0;
                            sudoku[m][j].nposib--;
                        }
                    }

                    /* Below body to find center point of corresponding quadrant */
                    int Ci=-1,Cj=-1;
                    if(i+1 == 1 || i+1 == 4 || i+1 == 7)
                        Ci = i+1;
                    else if(i==1 || i==4 || i==7)
                        Ci = i;
                    else if(i-1 == 1 || i-1 == 4 || i-1 == 7)
                        Ci = i-1;

                    if(j+1 == 1 || j+1 == 4 || j+1 == 7)
                        Cj = j+1;
                    else if(j==1 || j==4 || j==7)
                        Cj = j;
                    else if(j-1 == 1 || j-1 == 4 || j-1 == 7)
                        Cj = j-1;
                    /* Center point of quardrant is denoted by 'Ci' and 'Cj'    */

                    for(int m = Ci-1; m < Ci+2; ++m)
                    {
                        for(int n = Cj-1; n < Cj+2; ++n)
                        {
                            if(sudoku[m][n].posib[ sudoku[i][j].val -1 ] != 0)
                            {
                                sudoku[m][n].posib[ sudoku[i][j].val -1 ] = 0;
                                sudoku[m][n].nposib--;
                            }
                        }
                    }
                    /* Above nested loop accesses all values present in quadrant */

                goto start_cuz_values_have_changed;     // goes only if a value has been set
            }
        }
    }
}

void backup()
{
/**
    This set of code simply backs up
    the current sudoku so that guessing
    can be done                             **/

    // this results in save[][] being the same as sudoku
    for(int i = 0; i < 9; ++i)
    {
        for(int j = 0; j < 9; ++j)
        {
            save[i][j].val = sudoku[i][j].val;
            save[i][j].nposib = sudoku[i][j].nposib;
            for(int m = 0; m < 9; ++m)
                save[i][j].posib[m] = sudoku[i][j].posib[m];
        }
    }
}

```

```
199 void setsudoku()
200 {
201 /**
202     This set of code simply resets
203     the modulated sudoku so that guessing
204     can be done                              **/
205
206     // this results in sudoku[][] being the same as save[][]
207     for(int i = 0; i < 9; ++i)
208     {
209         for(int j = 0; j < 9; ++j)
210         {
211             sudoku[i][j].val = save[i][j].val;
212             sudoku[i][j].nposib = save[i][j].nposib;
213             for(int m = 0; m < 9; ++m)
214                 sudoku[i][j].posib[m] = save[i][j].posib[m];
215         }
216     }
217 }
218
219 void rfrequency()
220 {
221     set_posib();
222     singletons();   // directly calls these functions, hence eliminating the need to call them in main()
223
224     rowcheck:
225     int counter = 0;    // to check whether row has frequency change or not
226     // row-wise
227     for(int i = 0; i < 9; ++i) // traversers from row 1-9
228     {
229         int freq[9] = {0,0,0,0,0,0,0,0,0};
230         for(int j = 0; j < 9; ++j)
231         {
232             if(sudoku[i][j].nposib!=0) // if the values is not set already
233                 for(int k = 0; k < 9; ++k)
234                 {
235
236                     if(sudoku[i][j].posib[k]!=0)    // and if the possibility is non-zero
237                     {
238                         freq[k]++;
239                     }
240                 }
241         }
242         int n = 0;
243         for(int k = 0; k < 9; ++k)
244         {
245             if(freq[k]==1)
246             {
247                 n = k+1; // n holds value of number with 1 frequency
248                 break;
249             }
250         }
251         // If number with 1 frequency exists
252         if(n!=0)
253         {
254             ++counter;
255             for(int j = 0; j < 9; ++j)  // for every element in that row
256             {
257                 if(sudoku[i][j].nposib!=0)          // if value is already not present
258                     if(sudoku[i][j].posib[n-1]!=0)  // cuz that is value with frequency 1
259                     {
260                         // set val
261                         sudoku[i][j].val = n;
262                         sudoku[i][j].nposib = 0;
263                         for(int m = 0; m < 9; ++m)
264                         {
```

```
                        sudoku[i][j].posib[m] = 0;
                    }
                    break;
                }
            }
        }
    }
    if(counter!=0)
    {
        set_posib();    // if change has been made, call set_posib
        singletons();   // call singleton function, set singletons again cuz some new ones may be formed!
        goto rowcheck;  // restart row wise check
    }

    /**   ONCE ALL ROWS HAVE BEEN SET , START WORKING ON COLUMNS  **/

    int counter2 = 0;
    // col-wise
    for(int i = 0; i < 9; ++i) // traversers from col 1-9
    {
        int freq2[9] = {0,0,0,0,0,0,0,0,0};
        for(int j = 0; j < 9; ++j)
        {
            for(int k = 0; k < 9; ++k)
            {
                if(sudoku[j][i].nposib!=0) // if the values is not set already
                    if(sudoku[j][i].posib[k]!=0)    // and if the possibility is non-zero
                    {
                        freq2[k]++;
                    }
            }
        }
        int n2 = 0;
        for(int k = 0; k < 9; ++k)
        {
            if(freq2[k]==1)
            {
                n2 = k+1; // n2 holds value of number with ! frequency
                break;
            }
        }
        // If number with 1 frequency exists
        if(n2!=0)
        {
            ++counter2;
            for(int j = 0; j < 9; ++j)
            {
                if(sudoku[j][i].nposib!=0)          // if value is already not present
                    if(sudoku[j][i].posib[n2-1]!=0)  // cuz that is value with frequency 1
                    {
                        // set val
                        sudoku[j][i].val = n2;
                        sudoku[j][i].nposib = 0;
                        for(int m = 0; m < 9; ++m)
                        {
                            sudoku[j][i].posib[m] = 0;
                        }
                        break;
                    }
            }
        }
    }
    if(counter2!=0)
    {
        set_posib();    // if change has been made, call set_posib
        singletons();   // call singleton function, set singletons again cuz some new ones may be formed!
```

```
331         goto rowcheck;  // restart row wise check
332     }
333 
334     /**   ONCE ALL COLS HAVE BEEN SET , START WORKING ON QUADRANTS  **/
335 
336     // quardrant - vise
337     int counter3 = 0;
338 
339     // note that the loop only gives i = j= {1,4,7} which are quadrant centers
340     for(int i = 1; i < 8; i+=3)
341     {
342         for(int j = 1; j < 8; j+=3)
343         {
344             // for every box henceforth
345             int freq3[9] = {0,0,0,0,0,0,0,0,0};
346             for(int Ci = i-1; Ci<=i+1; ++Ci)
347             {
348                 for(int Cj = j-1; Cj<=j+1; ++Cj)
349                 {
350                     if(sudoku[Ci][Cj].nposib!=0)    // if the value has not been determined
351                     {
352                         for(int k = 0; k < 9; ++k)
353                         {
354                             if(sudoku[Ci][Cj].posib[k]!=0)  // if 'k'th posib exists,
355                                 freq3[k]++;
356                         }
357                     }
358                 }
359             }
360             int n3 = 0;
361             for(int k = 0; k < 9; ++k)
362             {
363                 if(freq3[k]==1)
364                 {
365                     n3=k+1;
366                     break;
367                 }
368             }
369 
370             if(n3!=0)       // if a frequency 1 value exists
371             {
372                 ++counter3;
373                 for(int Ci = i-1; Ci<=i+1; ++Ci)
374                 {
375                     for(int Cj = j-1; Cj<=j+1; ++Cj)
376                     {
377                         // every element in the quadrant
378                         for(int k = 0; k < 9; ++k)
379                         {
380                             if(sudoku[Ci][Cj].posib[n3-1] != 0) // if required box is located
381                             {
382                                 sudoku[Ci][Cj].val = n3;
383                                 sudoku[Ci][Cj].nposib = 0;
384 
385                                 for(int r = 0; r < 9; ++r)
386                                     sudoku[Ci][Cj].posib[r] = 0;
387                                 break;
388 
389                             }
390                         }
391                     }
392                 }
393             }
394         }
395     }
396     if(counter3!=0)
```

```
    {
        set_posib();    // if change has been made, call set_posib
        singletons();   // call singleton function, set singletons again cuz some new ones may be formed!
        goto rowcheck;  // restart row wise check
    }
}


void guess()
{
    cout << "initiating brute force algorithm \n";
    starting:
    int row=-1,col=-1,val=0;
    for(int i =0; i < 9; ++i)
    {
        for(int j = 0; j < 9; ++j)
        {
        // goes through every element
            if(sudoku[i][j].val==0)
            {
                row=i;
                col=j;
                for(int k = 0; k < 9; ++k)
                {
                    if(sudoku[i][j].posib[k]!=0)
                    {
                        val = sudoku[i][j].posib[k];
                        sudoku[i][j].val = val;
                        sudoku[i][j].nposib=0;
                        goto loop_stop;
                    }
                }
            }
        }
    }
    loop_stop:

    for(int k = 0; k<9; ++k)
        sudoku[row][col].posib[k]=0;

    rfrequency();

    // sudoku with a guess has been solved
    // loop then runs to see if it worked

    for(int i = 0; i < 9; ++i)
    {
        for(int j = 0; j < 9; ++j)
        {
        // for very element in the sudoku

            if(sudoku[i][j].val == 0)   // if not solved
            {
                // if no solution is possible
                // then make changes to save[][]
                // resetsudoku according to change
                if(sudoku[i][j].nposib==0)
                {
                    save[row][col].posib[val-1]=0;
                    save[row][col].nposib-=1;
                    save[row][col].val=0;       // just incase
                    setsudoku();
                }
                cout << '.';
                goto starting;
            }
```

```
        }
    }
    cout << "\nsuccessful brute force execution!\n";

}


// the final message!
void view()
{
    cout << endl << endl;
    cout <<" ######  ##     ## ########   #######  ##    ## ##     ## "<< endl;
    cout <<"##    ## ##     ## ##     ## ##     ## ##   ##  ##     ## "<< endl;
    cout <<"##       ##     ## ##     ## ##     ## ##  ##   ##     ## "<< endl;
    cout <<" ######  ##     ## ##     ## ##     ## ##### ##  ##     ## "<< endl;
    cout <<"      ## ##     ## ##     ## ##     ## ##  ##   ##     ## "<< endl;
    cout <<"##    ## ##     ## ##     ## ##     ## ##   ##  ##     ## "<< endl;
    cout <<" ######   #######  ########   #######  ##    ##  ####### "<< endl;
    cout << endl;
    cout <<" ######   #######  ##       ##     ## ######## ######## "<< endl;
    cout <<"##    ## ##     ## ##       ##     ## ##       ##     ## "<< endl;
    cout <<"##       ##     ## ##       ##     ## ##       ##     ## "<< endl;
    cout <<" ######  ##     ## ##       ##     ## ######   ######## "<< endl;
    cout <<"      ## ##     ## ##        ##   ##  ##       ##   ## "<< endl;
    cout <<"##    ## ##     ## ##         ## ##   ##       ##    ## "<< endl;
    cout <<" ######   #######  ########    ###    ######## ##     ## "<< endl;
    cout << endl
         << endl
         << endl
         << "              BBBB       \n"
         << "              B   B      \n"
         << "              BBBB  y  y \n"
         << "              B   B y  y \n"
         << "              BBBB   yyy \n"
         << "                     y \n"
         << "                  yyy  \n"
         << endl
         << endl
         << endl
         << "    #                                                  " << endl
         << "   # #   #####  #    # #  ####  #    # ###### #    # " << endl
         << "  #   #  #    # #    # # #      #    # #      #   #  " << endl
         << " #     # #####  ###### #  ####  ###### #####  ####   " << endl
         << " ####### #    # #    # #      # #    # #      #  #   " << endl
         << " #     # #    # #    # # #    # #    # #      #   #  " << endl
         << " #     # #####  #    # #  ####  #    # ###### #    # " << endl
         << endl
         << endl
         << endl;

}

int main()
{
    cout << endl;
    cout << "Welcome to the sudoku solver! \n";
    cout << endl;
    cout << "This program is specifically tailored to solve any valid sudoku you enter.\n";
    b:
    cout << endl;
    cout << "Please enter a valid sudoku for expected results : \n\n";
    char ch;
/****************************** INPUT *****************************************/

    cout << "************************" << endl;
    for(int i = 0; i < 9; ++i)
```

```
    {
        for(int j = 0; j < 9; ++j)
        {
            a:
            ch = getch();
            if(ch > '0' && ch <= '9')
            {
                sudoku[i][j].val = (int)ch - 48;
                cout << sudoku[i][j].val;
            }
            else if(ch=='\n' || ch=='\r')
            {
                sudoku[i][j].val = 0;
                cout << "-";
            }
            else
            goto a;
            if((j+1)%3==0)
                cout << " * ";
            cout << " ";
        }
        cout << endl;
        if((i+1)%3==0)
            cout << "*************************" << endl;
    }

/***************************** DISPLAY ********************************************/

    cout << "\nThank you for the input..." << endl;
    cout << "Please check if this is the correct sudoku : \n\n";

    cout << "*************************************" << endl;
    for(int i = 0; i < 9; ++i)
    {
        for(int j = 0; j < 9; ++j)
        {
            if(sudoku[i][j].val!=0)
                cout << sudoku[i][j].val << "  ";
            else cout << "-  ";

            if((j+1)%3==0)
                    cout << " *   ";
        }
        cout << endl;
        if((i+1)%3==0)
            cout << "*************************************" << endl;
    }
    cout << endl << "Is the correct sudoku (y/n) : " ;
    cin >> ch;
    if(ch=='N' || ch == 'n')
        goto b;
    else
    cout << "the program shall now start solving the sudoku \n\n";

/********************************** SOLUTION ***************************************/

    rfrequency();
    backup();

    cout << endl;

/****************** Call for guessing *************************/

for(int i = 0; i < 9; ++i)
{
    for(int j = 0; j < 9; ++j)
```

```
    {
        if(save[i][j].val==0)
        {
            guess();
            backup();
            goto loop_term;
        }
    }
}
loop_term:

/*****************************************************/

        cout << "\n\nAnd the complete solved sudoku is : \n\n";

    // display after brute force solution
        cout << " ***************************************" << endl;
        for(int i = 0; i < 9; ++i)
        {
            cout << " *   ";
            for(int j = 0; j < 9; ++j)
            {
                if(sudoku[i][j].val!=0)
                    cout << sudoku[i][j].val << "  ";
                else cout << "-  ";
                if((j+1)%3==0)
                    cout << " *   ";
            }
            cout << endl;
            if((i+1)%3==0)
                cout << " ***************************************" << endl;
        }

/*****************************************************/

    cout << "\n\nDo you wish to save this sudoku solution(y/n) : ";
    cin >> ch;
    if(ch=='y' || ch == 'Y')
    {
        ofstream solution("solutions.txt", ios_base::app | ios::out);
        cout << "What do you want this solution to be named as : ";
        char puzzle_name[10];
        cin >> puzzle_name;
        solution << endl << puzzle_name << endl;
        solution << "*************************************\n";
        for(int i = 0; i < 9; ++i)
        {
            for(int j = 0; j < 9; ++j)
            {
                solution << sudoku[i][j].val << "  ";
                if((j+1)%3==0)
                    solution << " *   ";
            }
            solution << endl;
            if((i+1)%3==0)
                solution << "*************************************\n";
        }

        solution.close();
        cout << endl
             << "Solution successfully appended to \"solutions.txt\"."
             << endl << endl;
        getch();
    }

    cout << "\n\n\n\n\nThank you for using this programme and i hope it impressed you!\n\n\n" << endl <<
```

```
endl;
661       view();
662
663       return 0;
664   }
```