```
#include<iostream>

#include<string.h>

#include<vector>


#define LIVE 1

#define DEAD 0


using namespace std;


class Cell

{

        private:

                int iState;

                string strName;


        public:

                // Default Constructor

                Cell()

                {

                        iState = DEAD;

                        strName = "default";

                }


                // Parameterized constructor

                Cell(string strName, int iState)
```

```
{
        this->iState = iState;
        this->strName = strName;
}

bool isAlive()
{
        if(this->iState == DEAD)
        {
                return false;
        }
        return true;
}

string getName()
{
        return this->strName;
}

int getState()
{
        return this->iState;
}

bool setState(int iState)
```

```
                {
                        if(iState > LIVE || iState < DEAD)
                        {
                                return false;
                        }

                        this->iState = iState;

                        return true;
                }

                bool setName(string strName)
                {
                        if(strName == "")
                        {
                                return false;
                        }

                        this->strName = strName;

                        return true;
                }
};

class Grid
```

```
{
    private:
        int iRow, iCol;
        int iRowOffset, iColOffset;

        vector<vector<Cell>> vctGrid;
        vector<Cell> vctRow;

        void FillGrid()
        {
            Cell objCell;

            vector<Cell> vctTemp(iCol, objCell);

            for(int i = 0; i < iRow; ++i)
            {
                vctGrid.push_back(vctTemp);
            }
        }

        void DisplayRow(vector<Cell> vct)
        {
            for(auto x : vct)
            {
                cout<<x.getState()<<x.getName()<<"\t";
        }
        cout<<"\n";
    }

    int GetLiveNeighbors(int iRow, int iCol)
    {
        int iLiveCount = 0;
        vector<vector<int>> vctNeighbours = { {-1,0},{-1,1},{0,1},{1,1},{1,0},{1,-1},{0,-1},{-1,-1} };

        /*
            Positions of all neighbours of given (row,col)
                (i-1,j-1)   (i-1,j) (i-1,j+1)
                (i,j-1)         (i,j)     (i,j+1)
                (i+1,j-1)   (i+1,j)    (i+1,j+1)
        */
        for(auto vct : vctNeighbours)
        {
            int x = vct[0] + iRow;
            int y = vct[1] + iCol;

            if(x >= 0 && x < this->iRow && y >= 0 && y < this->iCol &&
vctGrid[x][y].isAlive())
            {
                iLiveCount++;
            }
        }
```

```
		return iLiveCount;
	}

public:
	// Default Constructor
	Grid()
	{
		this->iRow = 100;
		this->iCol = 100;
		iRowOffset = 0;
		iColOffset = 0;
		// Create grid with default max size
		FillGrid();
	}

	// Parameterized constructor
	Grid(int iRow, int iCol)
	{
		this->iRow = iRow;
		this->iCol = iCol;
		iRowOffset = 0;
		iColOffset = 0;
		// Create the grid of given size with default values
		// All cells are DEAD as default
```

```
                        FillGrid();
                }

                bool InsertCell(string strName, int iState)
                {
                        if((strName == "") || (iState < DEAD) || (iState > LIVE) || (iColOffset == iCol) ||
(iRowOffset == iRow))
                        {
                                return false;
                        }

                        // Create a new Cell
                        Cell objCell(strName, iState);

                        // Insert the newly created cell in grid
                        vctGrid[iRowOffset][iColOffset] = objCell;

                        // Keep inserting into the same row until the row is filled
                        if(iColOffset != iCol - 1)
                        {
                                iColOffset++;
                        }
                        else
                        {
                                // When a row is filled move to next row
                                iRowOffset++;
                }

                        // Reset column offset to again start from 0th position of current
row
                        iColOffset = 0;
                }

                return true;
        }

        int GetCellState(int iRow, int iCol)
        {
                if((iRow >= this->iRow) || (iCol >= this->iCol))
                {
                        return -1;
                }

                return vctGrid[iRow][iCol].getState();
        }

        void DisplayGrid()
        {
                cout<<"\n";
                for(auto vct : vctGrid)
                {
                        DisplayRow(vct);
                }
        }
```

```
void NextState()
{
    Cell ob;
    vector<vector<Cell>> vctNext(iRow, vector<Cell>(iCol, ob));

    for(int i = 0; i < iRow; ++i)
    {
        for(int j = 0; j < iCol; ++j)
        {
            int iLiveCount = GetLiveNeighbors(i,j);

            vctNext[i][j].setName(vctGrid[i][j].getName());

            if(vctGrid[i][j].isAlive() && ((iLiveCount < 2) || (iLiveCount > 3)))
            {
                vctNext[i][j].setState(DEAD);
            }
            else if(vctGrid[i][j].isAlive() && (iLiveCount == 2 || iLiveCount ==
3))
            {
                vctNext[i][j].setState(GetCellState(i,j));
            }
            else if(!vctGrid[i][j].isAlive() && (iLiveCount == 3))
            {
                vctNext[i][j].setState(LIVE);
            }


        }

    }


    vctGrid = vctNext;

}


int SearchCell(string strName)

{

    if(strName != " ")

    {

        int i = 0, j = 0;


        for(i = 0; i < iRow; ++i)

        {

            for(j = 0; j < iCol; ++j)

            {

                if(vctGrid[i][j].getName() == strName)

                {

                    return GetCellState(i,j);

                }

            }

        }

    }
```

```
                        return -1

                    }

        };


        int main()

        {

                    int iChoice = -1

                    int iRow = 0, iCol = 0;

                    int iState = 0, iRet  = 0;

                    string strName;

                    char szState[6];


                    Grid *objGrid = NULL;


                    while(iChoice != 0)

                    {

                                cout<<"\n1) Create Cell Grid\n";

                                cout<<"2) Generate next state\n";

                                cout<<"3) Search Cell by name\n";

                                cout<<"4) Display Cell Grid\n";

                                cout<<"0) Exit\n";

                                cout<<"Enter Choice: ";

                                cin>>iChoice;
```

```
switch(iChoice)
{
        case 1 cout<<"\nEnter size of grid:\n";
                        cout<<"Rows: ";
                        cin>>iRow;
                        cout<<"Columns: ";
                        cin>>iCol;

                        objGrid = new Grid(iRow,iCol);

                        cout<<"-----------Please enter the data about cells------------\n";
                        for(int i = 0; i < iRow*iCol; ++i)
                        {
                                cout<<"\n---Cell "<<i+1<<" Data---";
                                cout<<"\nEnter Name: ";
                                cin>>strName;
                                cout<<"\nEnter State(Dead or Alive): ";
                                cin>>szState;

                                if(strcasecmp(szState, "dead") == 0)
                                {
                                        iState = DEAD;
                                }
                                else if(strcasecmp(szState, "alive") == 0)
                                {
                                        iState = LIVE;
                                }
                                else
                                {
                                        cout<<"\nPlease enter a valid state\n";
                                        --i;
                                        continue;
                                }

                                // Insert the cell into the grid
                                objGrid->InsertCell(strName, iState);
                        }
                        break;

                case 2: if(NULL == objGrid)
                        {
                                cout<<"\nError: No grid found\nPlease create a grid
first\n\n";
                        }
                        else
                        {
                                objGrid->NextState();
                                cout<<"\n---------The next state of the Cells---------\n";
                                objGrid->DisplayGrid();
                        }
                        break;
```

```
case 3: cout<<"\nEnter cell name to search: ";
        cin>>strName;

        iRet = objGrid->SearchCell(strName);

        if(iRet == -1)
        {
            cout<<"\nCell not found, Please enter valid name\n";
        }
        else
        {
            cout<<"\nCell Found!\nCell Name: "<<strName;
            if(iRet == DEAD)
            {
                cout<<"\nCell State: DEAD\n";
            }
            else
            {
                cout<<"\nCell State: ALIVE\n";
            }
        }
        break;

case 4: if(NULL == objGrid)
```

```
                                        {
                                                cout<<"\nError: No grid found\nPlease create a grid
first\n\n";
                                        }
                                        else
                                        {
                                                cout<<"\n---------The Cell Grid---------\n";
                                                objGrid->DisplayGrid();
                                        }
                                        break;

                        case 0: cout<<"\nThankyou for using our application\n";
                                        delete objGrid;
                                        break;

                        default: cout<<"\nPlease enter a valid input\n";
                                         break;
                }
        }

        /*
        Grid *objGrid = new Grid(4,3);

        objGrid->InsertCell("s", 0);
        objGrid->InsertCell("w", 1);
        objGrid->InsertCell("m", 0);
```

```
objGrid->InsertCell("b", 0);
objGrid->InsertCell("q", 1);
objGrid->InsertCell("e", 1);

objGrid->InsertCell("k", 1);
objGrid->InsertCell("q", 0);
objGrid->InsertCell("i", 1);

objGrid->InsertCell("a", 0);
objGrid->InsertCell("l", 1);
objGrid->InsertCell("p", 1);

objGrid->DisplayGrid();
cout<<endl;

objGrid->NextState();
objGrid->DisplayGrid();
cout<<endl;


objGrid->NextState();
objGrid->DisplayGrid();
cout<<endl;
*/
```

```
        return 0;
    }
```