Submit a report(in pdf format) of up to 25-30 pages, including problem statement, related work, detailed design including code, output screen shots , Future improvements and bibliography. One report per team is sufficient.

~~~~~~~~~~~~~~~~~~~~ GAME OF LIFE~~~~~~~~~~~~~~~~~~~~

This project is basically about the 'survival and death of the cells' , depending upon the several conditions provided.
And this game is all about the formation of dead cells (or) the destruction of live cells which continuously occur depending upon the conditions of a particular live cell or dead cell.

The **Game of Life**, also known simply as **Life**, is a cellular automation devised by the British mathematician John Horton Conway in 1970.

The "game" is actually a zero-player game, meaning that its evolution is determined by its initial state, needing no input from human players . One interacts with the Game of Life by creating an initial configuration and observing how it evolves.

RULES:

A cell **C** is represented by a **1** when alive, or **0** when dead(in this introduction), in An m x m square array of cells.

We calculate **N** - the sum of live cells in C's eight location neighbourhood then cell C is alive or dead in the next generation based on the following table:

C	**N**	**new C**
1	0,1	-> 0 # Lonely
1	4,5,6,7,8	-> 0 # Overcrowded
1	2,3	-> 1 # Lives
0	3	-> 1 # It takes three to give birth!
0	0,1,2,4,5,6,7,8	-> 0 # Barren
~~~~~~~~~~~~~~~~~~~~

Assume cells beyond the boundary are always dead

1.If a live cell is lonely (see the above chart) it dies.

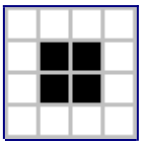
2.If a live cell has two or three live neighbours ,it lives on to the next generation.

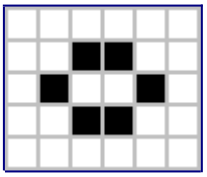
3.If a live cell is overcrowded (see the above chart) it dies.

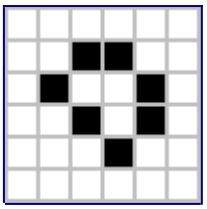
4. If a dead cell has exactly three live neighbours it becomes a living cell else it will be a dead cell.

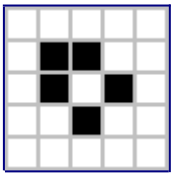
PATTERNS

Patterns of living cells...

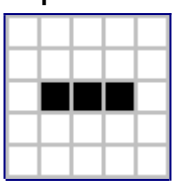
Block 

Beehive 

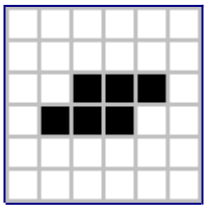
Loaf 

Boat 

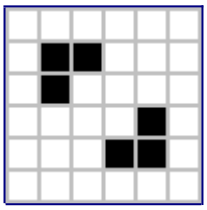
Examples of some oscillating patterns that never stops.

Blinker (period 2) 

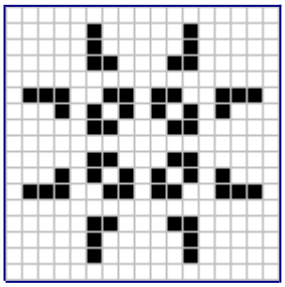
Toad (period 2)



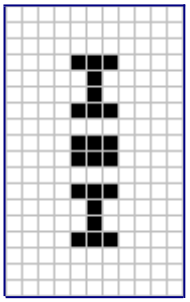
Beacon (period 2)



Pulsar (period 3)



Pentadecathlon
(period 15)



DESCRIPTION OF CODE:

Functions Used:

void srand(unsigned int seed); //pseudo random genarator

int main_menu(); //main menu function

int machine_game_menu(); //machine menu function

int human_game_menu(); //human menu function

void end_game(); //end game function

void load_random(TableType table); //generate random pattern

```
void user_input(TableType table);              //generate user pattern

int neighbour_value(TableType table, int row, int col);
// value of neighbour

int neighbour_count(TableType table, int row, int col);
// counts the number of neighbour

void calculate(TableType table);        // calculates according to the rules

void print_user_pattern(TableType table);
// prints the generations of user

void print_machine_pattern(TableType table);
// print the generations of machine

void sample_menu()                      // menu of different patterns

void sample_input(TableType table)
// prints the some beautiful outcomes
```

we used the following libraries

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
```

>>then we defined the following

```
#define HEIGHT 40  (defines the row of the table)
#define WIDTH 40 (defines the columns of the table)
#define LIVE_CELL 1
#define DEAD_CELL 0
```

We will explain our project by considering the above functions.....

1.Main function

```
int main()
{
  int opt1, opt2, opt3, opt4, k = 1;
```

```
TableType table;
srand ( time ( NULL ) );


      do
      {
          opt1 = main_menu();
          if (opt1 == 1)
          {
                do
                {
                   opt2 = machine_game_menu();
                         if(opt2 == 1)
                         {
                            printf("@Machine Destiny Game@\n");
                             load_random( table );
                             do
                             {
                                 print_machine_pattern( table );
                                 calculate( table );
                                 usleep(450000);
                             }while(1);
                         }
                         if(opt2 > 2)
                         {
                                 printf("\t\t-WRONG INPUT-\n\n");
                         }
                }while(opt2 > 2);
          }
          else if (opt1 == 2)
          {
             do
             {
                   opt3 = human_game_menu();
                         if(opt3 == 1)
                         {
                             printf("@Human Destiny Game@\n");
                             printf("Enter Pattern\n");
                             user_input( table );
                             do
                             {
                                 print_user_pattern( table );
```

```
                        calculate( table );
                        usleep(900000);
                    }while(1);
                }
                if(opt3 > 2)
                {
                        printf("\t\t-WRONG INPUT-\n\n");
                }
            }while(opt3 > 2);
        }
        else if(opt1 == 3)
        {

           sample_input(table);
                    do
                    {
                        print_user_pattern( table );
                        calculate( table );
                        usleep(450000);
                    }while(1);


        }

        else if (opt1 == 4)
        {
            end_game();
            break;
        }
        else
        {
            printf("\t\t-WRONG INPUT-\n\n");
        }
    }while( opt1 > 3 || opt2 == 2 || opt3 == 2 || opt4 == 2 );

    return 0;
}
```

>>In this main function we declared four variables to choose the options from the functions(main_menu,machine_menu,user_menu,sample_menu)

>>we declared a variable table of type TableType
>>when we are using a load_random function it gives the same value every time when we run it.So to get a different return value from the load_random function every time we run the program we use srand(time(null)) .For that we use the time.h librarary.
>>Using the do-while loop we give the user the options to choose what type of game he wants to play
the choices in main menu are
i)machine destiny game
j)human destiony game
k)sample input

MACHINE DESTINY GAME

>>we will pass TableType array into load_random() function which will be defined later and we used an infinite do-while loop and used the following functions
i)print_machine_pattern
j)calculate
and we used the function usleep() which will print the output with a certain amount of time delay as per given (usleep() functio uses the unistd.h library)

HUMAN DESTINY GAME

>>here we used user_input() function which will be defined below and after that the same functions as in machine destiny game are used like:
i)print_user_pattern
j)calculate
and we used the function usleep() which will print the output with a certain amount of time delay as per given (usleep() functio uses the unistd.h library)

SAMPLE INPUT

>>in this we defined the patterns previously in the sample_input() function and used the functions
i)print_user_pattern
j)calculate
k)usleep()

```
int machine_game_menu()
{
    int opt2;
    printf("\n\t\t=======================\n");
    printf("\t\t@ Machine Destiny Game @\n");
    printf("\t\t=======================\n");
```

```
	printf("In this game, computer will generate the pattern of cell
randomly for you.\n");
	printf("All you have to do is sit back and watch the cell progress.\n");
	printf("Please choose option below to proceed and have fun!\n");
	printf("1.Start Game\n");
	printf("2.Main Menu\n");
	printf("\tYour Selection: ");
	scanf ("%d", &opt2);
	int i=0;
	  while(i<50)
	  {
		printf("\n");
	  i++;
	  }
	return opt2;
}
```

in the above function we have displayed machine destiny menu with two choices

i)start game

j)main menu

```
void load_random(TableType table)
{
	int y,z;
	for(y=0;y<HEIGHT;y++)
	{
		for(z=0;z<WIDTH;z++)
		{
			table[y][z]=DEAD_CELL;
		}
	}

	int x, row, col, i;
	x = ( rand() % 1000 ) + 10;
	for(i = 1;i <= x; i++)
	{
		row = ( rand() % 60) + 4;
		col = ( rand() % 60 ) + 4;
		table[row][col] = LIVE_CELL;
	}
}
```

>>in this load_random() function we intitialized each element in table variable to DEAD_CELL (which is 0) to eliminate junk values and then we run the for loop x times whixh give x no. Of coordinates
>>as you can see x=rand()%1000+10 which means the first part will return a random value ranging from 0 to 1000. the least case is it returns 0 so we are adding 10 to it.
>>we will run another for loop x times and we will get the cordinates using the rand function.and we will intialize that element to a living cell. This doesn't mean that it gives x different coordinates because some may overlap while using the rand function
>>rand() function uses stdlib library

print_machine_pattern

```
void print_machine_pattern(TableType table)
{
	int row, col;

//	======================
//	@Machine Destiny Game@
//	======================

	printf("\033[H");
	for ( row = 0; row < HEIGHT; row++ )
	{
		for ( col = 0; col < WIDTH; col++ )
		{
			if (table[row][col] == LIVE_CELL)
			{
				printf("\033[07m  \033[0m\033[37m\033[40;01m");
			}
			else
			{
				printf("  ");
			}
		}
		printf("\033[E");
	}
	fflush(stdout);
}
```

>>we used eco commands in this function like

i)\033[H – It brings the cursor to the top left position i.e., to the initial position
j)\033[07m – it reverses the background color and the text color
k)\033[0m – it is used for printing the default colors(neither bold nor dim)
l)\033[37m – it gives text color , white is refered to 37
m)\003[40;01m – it gives background color , black is referred to 40 and ;01 indicates bold color(dark)
n)\033[E – to go to the next line
o)fflush() - it flushes out that is it removes buffer values in stdout(standard output).initially we didnot use fflush which resulted in some buffer values

user_input

```
void user_input(TableType table)
{
	int y,z;
	for(y=0;y<HEIGHT;y++)
	{
		for(z=0;z<WIDTH;z++)
		{
			table[y][z]=DEAD_CELL;
		}
	}

	int i,j;
	int n;
	int height, width;


	printf("Enter the amount of initial organisms: ");
	scanf("%d", &n);
	for (i = 0; i < n; i++)
	{
		printf("Enter dimensions (x y) where organism %d will live: ", i + 1);
		scanf("%d %d", &height, &width);
		table[height][width] = LIVE_CELL;
	}
	printf("@ Your input will look like as follows @");
	for(i = 0; i < HEIGHT; i++)
	{
		for(j = 0; j < WIDTH; j++)
```

```
            {
                if( table[i][j] == LIVE_CELL )
                    {printf(" * ");}
                else
                    printf("  ");
            }
            printf("\n");
        }
        int w=0;
        while(w<20)
        {
            printf("\n");
        w++;
        }
}
```

>>in this user_input() function first we ask the user if he want to see the scale of coordinates of the plane we defined
>>then we use simple for loops and take input from the user

neighbour_value

```
int neighbour_value(TableType table, int row, int col)
{
        if (row < 0 || row >= HEIGHT
            || col < 0 || col >= WIDTH
            || table[row][col] != LIVE_CELL )
        {
            return 0;
        }
        else
        {
            return 1;
        }
}
```

>> in this function we take variable table , row , column as parameters> this function will check the neighbouring cells and returns 1 if it is alive otherwise it returns zero

neighbour_count

```
int neighbour_count(TableType table, int row, int col)
```

```
{
	int neighbour = 0;

	neighbour += neighbour_value(table, row - 1, col - 1);
	neighbour += neighbour_value(table, row - 1, col);
	neighbour += neighbour_value(table, row - 1, col + 1);
	neighbour += neighbour_value(table, row, col - 1);
	neighbour += neighbour_value(table, row, col + 1);
	neighbour += neighbour_value(table, row + 1, col - 1);
	neighbour += neighbour_value(table, row + 1, col);
	neighbour += neighbour_value(table, row + 1, col + 1);

	return neighbour;
}
```

>>it returns the no. Of living cells around the cell which we give as input

calculate

```
void calculate(TableType table)
{
	TableType tableB;
	int neighbour, height, width;

	for (height = 0; height < HEIGHT; height++)
	{
		for (width = 0; width < WIDTH; width++)
		{
			neighbour = neighbour_count(table, height, width);
			if (neighbour==3)
			{
				tableB[height][width] = LIVE_CELL;
			}
			else if (neighbour == 2 && table[height][width] ==
LIVE_CELL)
			{
				tableB[height][width] = LIVE_CELL;
			}
			else
			{
				tableB[height][width] = DEAD_CELL;
			}
		}
	}
```

```
	for (height = 0; height < HEIGHT; height++)
	{
		for (width = 0; width < WIDTH; width++)
		{
			table[height][width] = tableB[height][width];
		}
	}
}
```
>>in this function we will make use of neighbour_count()
>>if the neighbour_count() is 3 we will make that cell alive and then we use an other if to check if the taken cell is a living cell and if the neighbour_count is 2 or 3 we will make it alive otherwise we make it a deadcell

print_user_pattern

```
void print_user_pattern(TableType table)
{
	int row, col;

//	======================
//	@ Human Destiny Game @
//	======================

	printf("\033[H");
	for ( row = 0; row < HEIGHT; row++ )
	{
		for ( col = 0; col < WIDTH; col++ )
		{
			if (table[row][col] == LIVE_CELL)
			{
				printf("\033[07m  \033[0m\033[37m\033[40;01m");
			}
			else
			{
				printf("  ");
			}
		}
		printf("\033[E");
	}
	fflush(stdout);
}
```

>>this function is same that as of print_machinepattern() function.the same commands are used

sample_menu()

```
 int sample_menu()
{
	printf("1.Figure eight\n2.infinite growth line\n3.Tumbler\
n4.infinite(horizantal)\n5.R-pentimino\n6.DieHardn");
	int i;
	printf("\tChoose number so that you could observe beautiful
patterns: ");
	scanf("%d",&i);
return i;
}
```

>>the above function is called if we want to implement the sample inputs
>>in this we will take the choice of the user

sample_input

```
void sample_input(TableType table)
{
	int t,i,j;
	for(i=0;i<HEIGHT;i++)
	{
		for(j=0;j<WIDTH;j++)
		{
			table[i][j]=0;
		}
	}

	t= sample_menu();
	if(t==1)
	{
		for(i=7;i<10;i++)
		{
			for(j=7;j<10;j++)
			{
			    table[i][j]=LIVE_CELL;
			}
		}
		for(i=10;i<13;i++)
		{
			for(j=10;j<13;j++)
			{
```

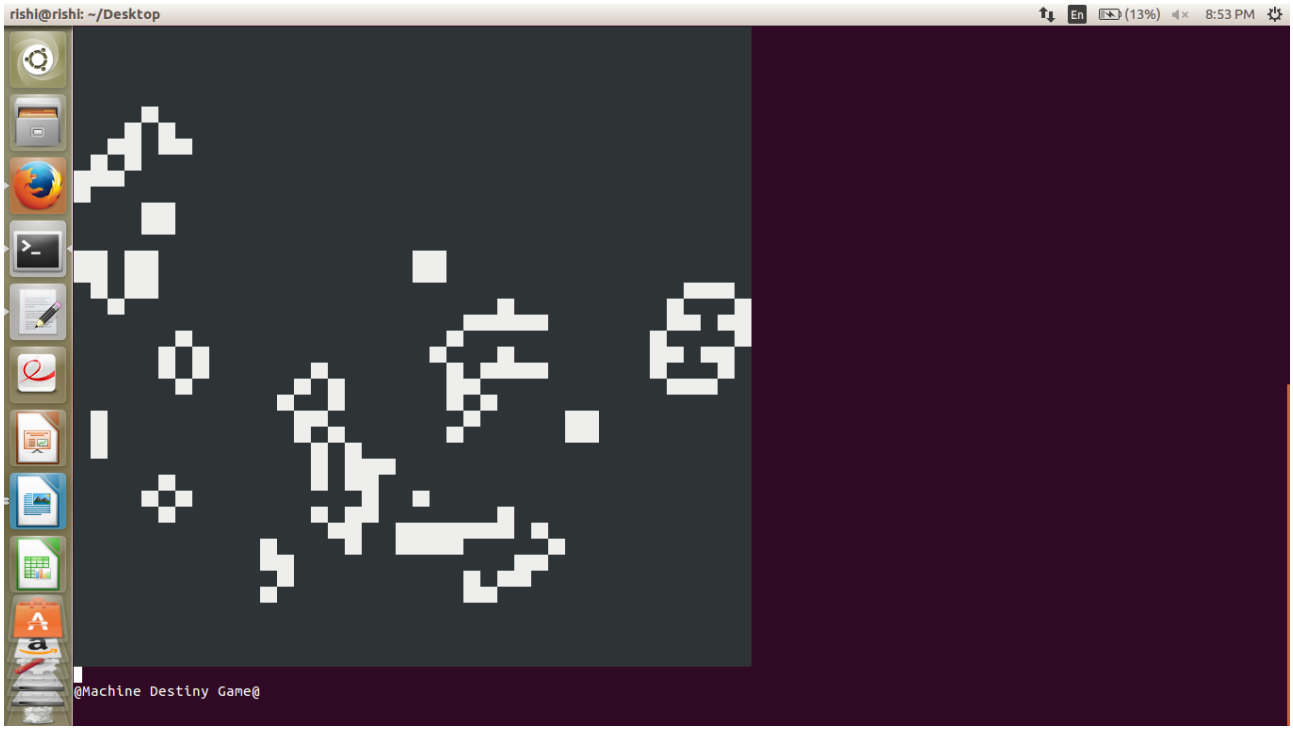
```
                table[i][j]=LIVE_CELL;
            }
        }
}
if(t==2)
{
        for(j=10;j<30;j++)
        {
          table[j][10]=LIVE_CELL;
        }
}
if(t==3)
{

        table[1][2]=LIVE_CELL;
        table[2][1]=LIVE_CELL;
        table[3][1]=LIVE_CELL;
        table[2][3]=LIVE_CELL;
        table[3][4]=LIVE_CELL;
        table[4][3]=LIVE_CELL;
        table[5][3]=LIVE_CELL;
        table[5][4]=LIVE_CELL;
        table[5][6]=LIVE_CELL;
        table[5][7]=LIVE_CELL;
        table[4][7]=LIVE_CELL;
        table[3][6]=LIVE_CELL;
        table[3][9]=LIVE_CELL;
        table[2][7]=LIVE_CELL;
        table[2][9]=LIVE_CELL;
        table[1][8]=LIVE_CELL;

}
if(t==4)
  {     int j;

        for(j=10;j<30;j++)
        {
          table[10][j]=LIVE_CELL;
        }
}
if(t==5)
{
```

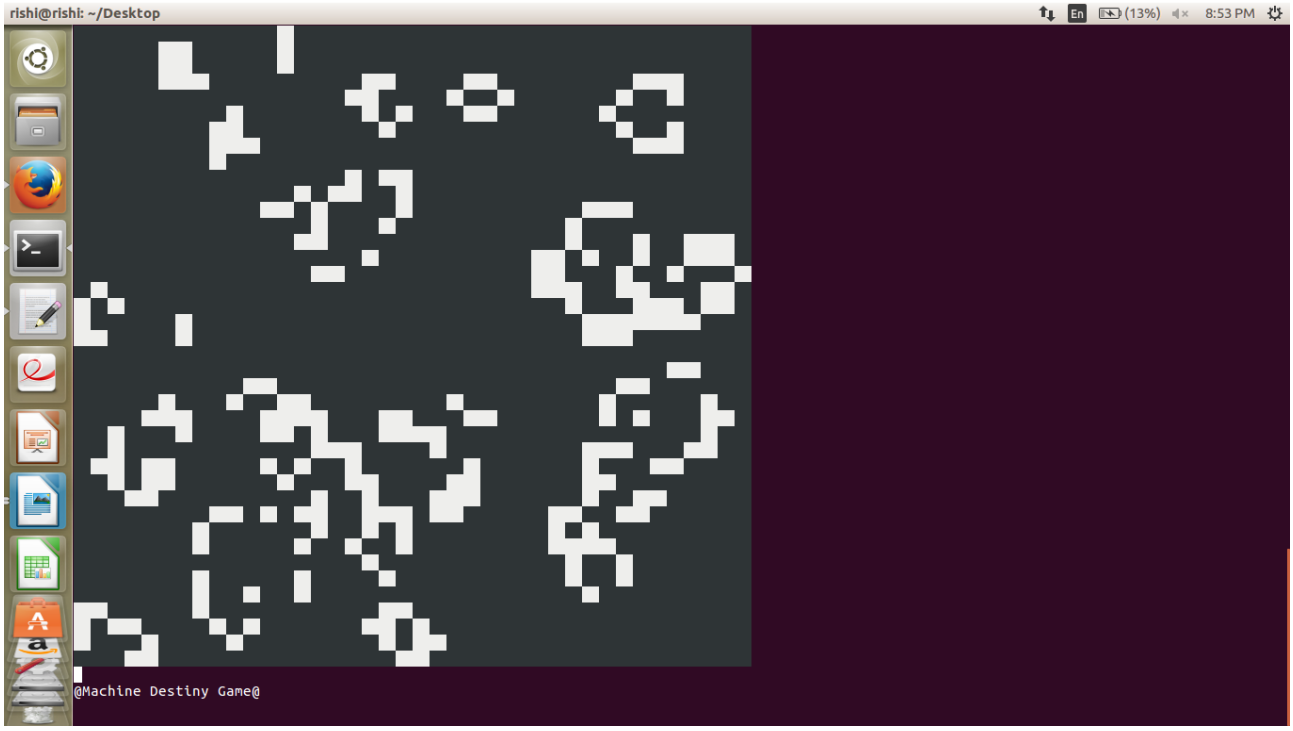
```
            table[14][10]=LIVE_CELL;
            table[14][11]=LIVE_CELL;
            table[15][9]=LIVE_CELL;
            table[15][10]=LIVE_CELL;
            table[16][10]=LIVE_CELL;
        }
        if(t==6)
        {
            table[15][5]=LIVE_CELL;
            table[15][6]=LIVE_CELL;
            table[16][6]=LIVE_CELL;
            table[16][10]=LIVE_CELL;
            table[14][11]=LIVE_CELL;
            table[16][11]=LIVE_CELL;
            table[16][12]=LIVE_CELL;
        }
        int u=0;
        while(u != 100)
        {
                printf("\n");
         u++;
        }
}
```

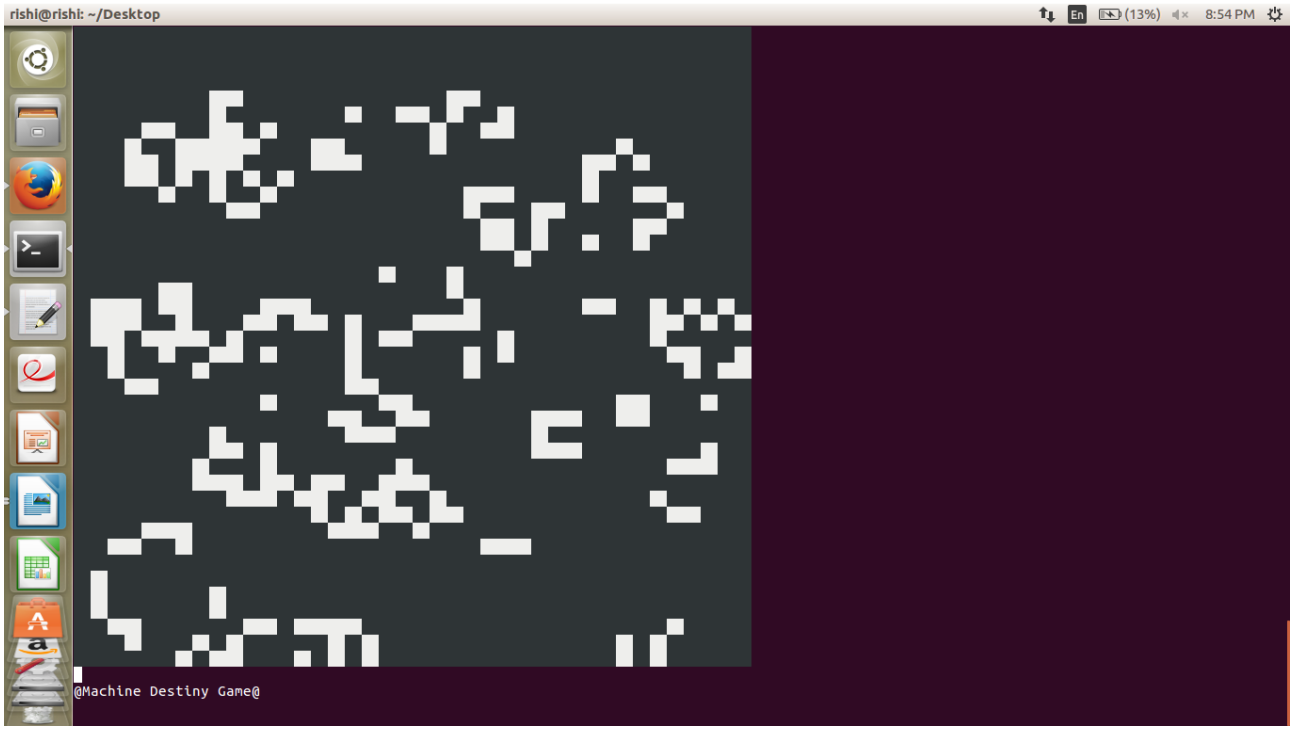
>>as per the option choosen by the user , we have given sample inputs to every sample pattern . That will be implemented

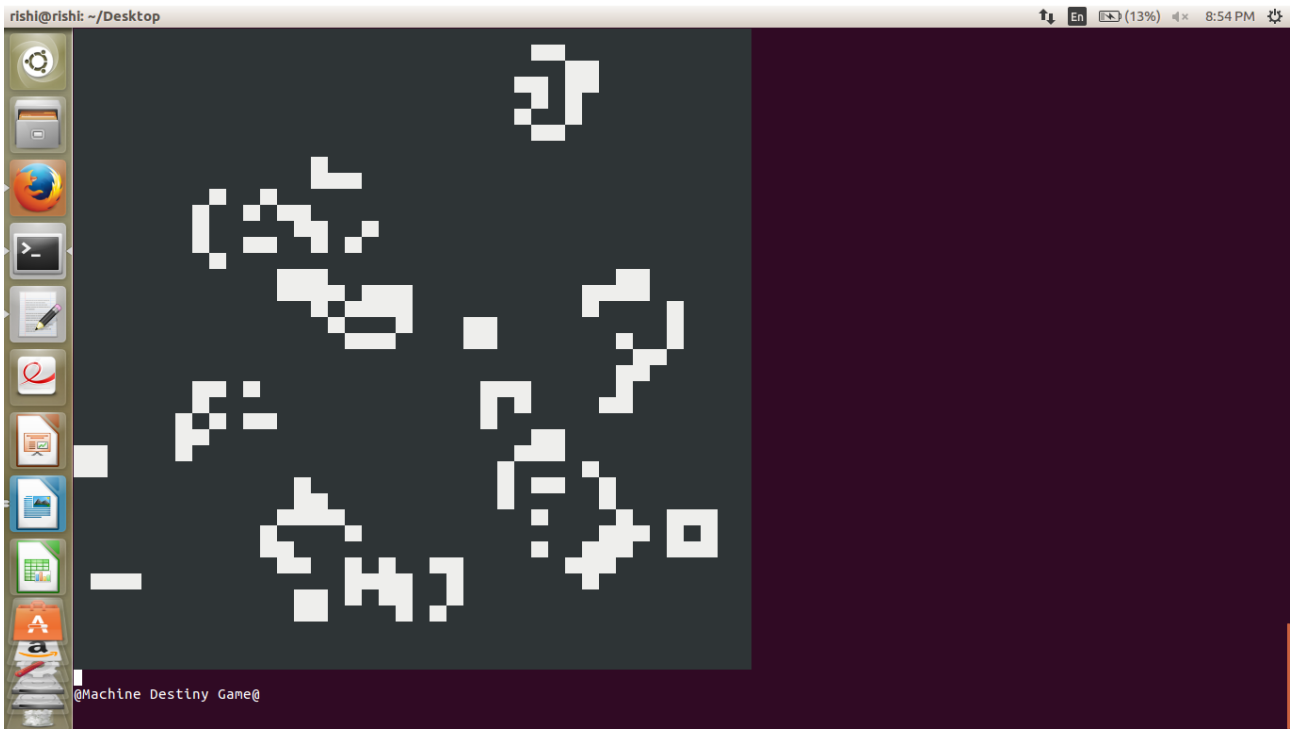
SCREEN SHOTS:

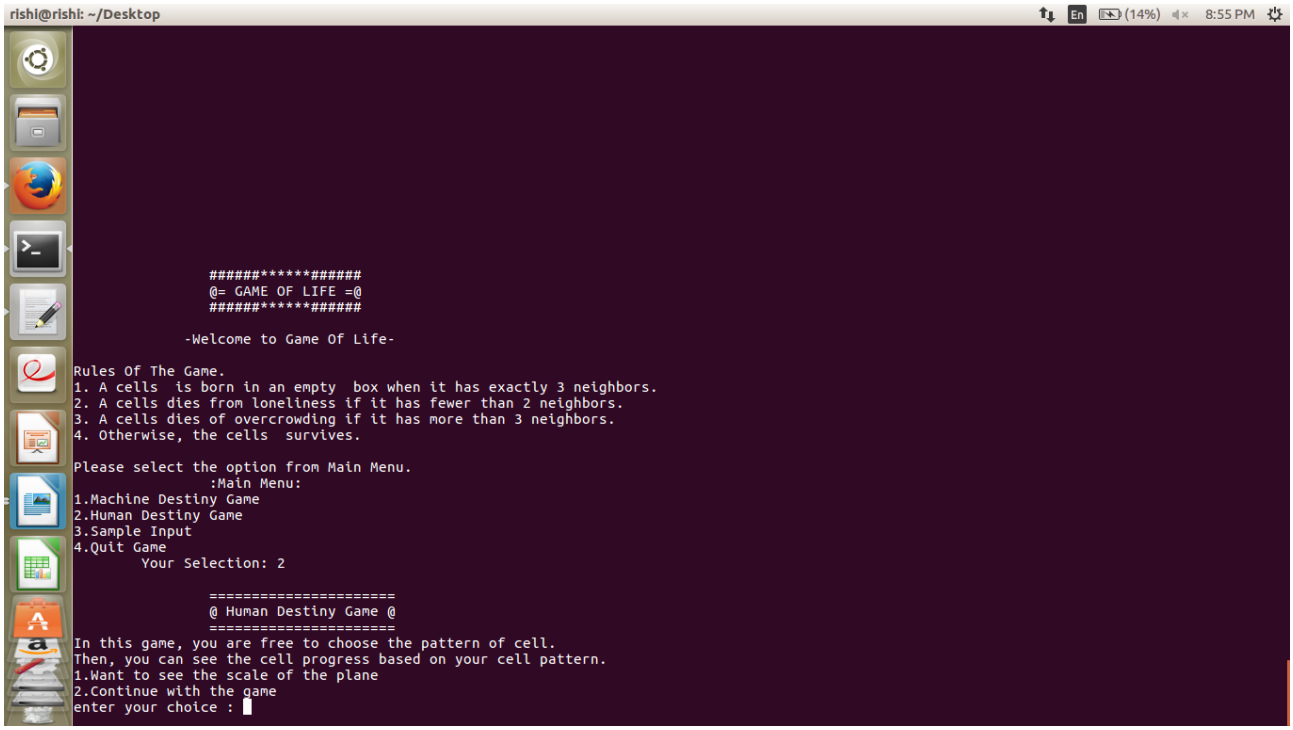
Following are some screen shots...

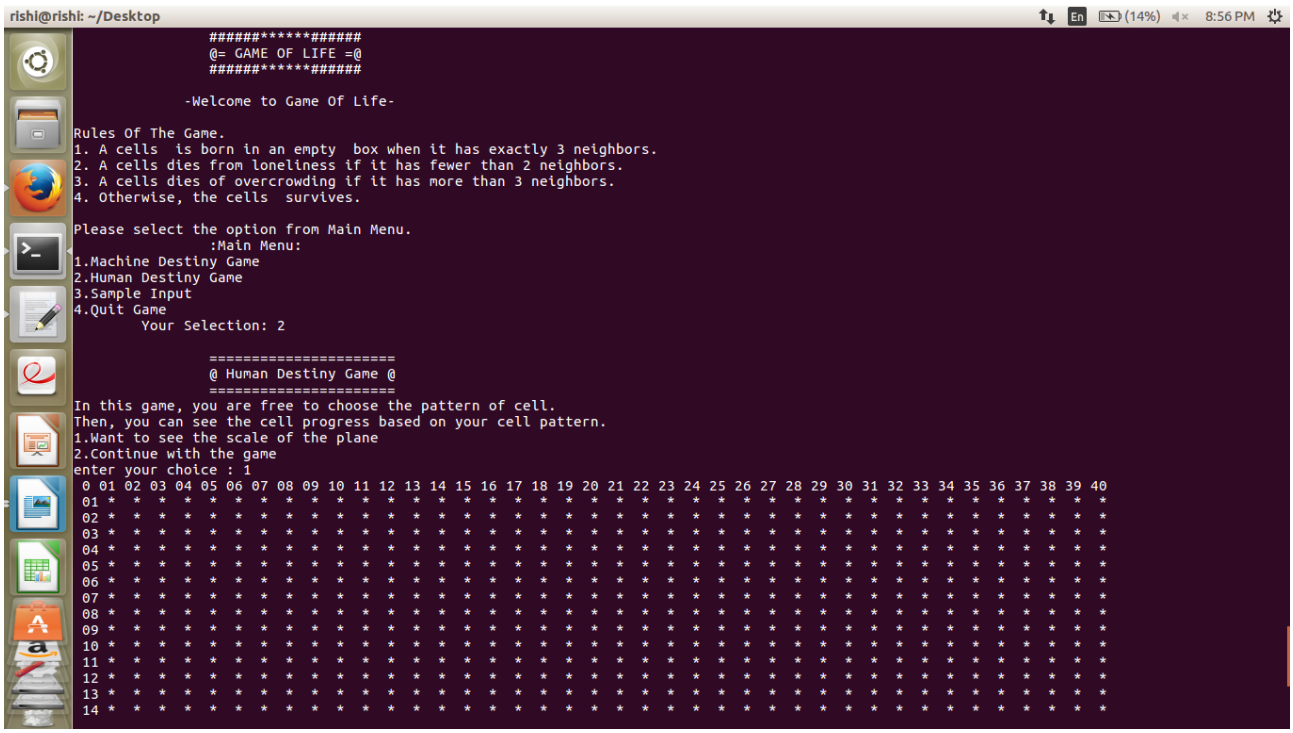

rishi@rishi: ~/Desktop
En
(13%)
8:53 PM
@Machine Destiny Game@

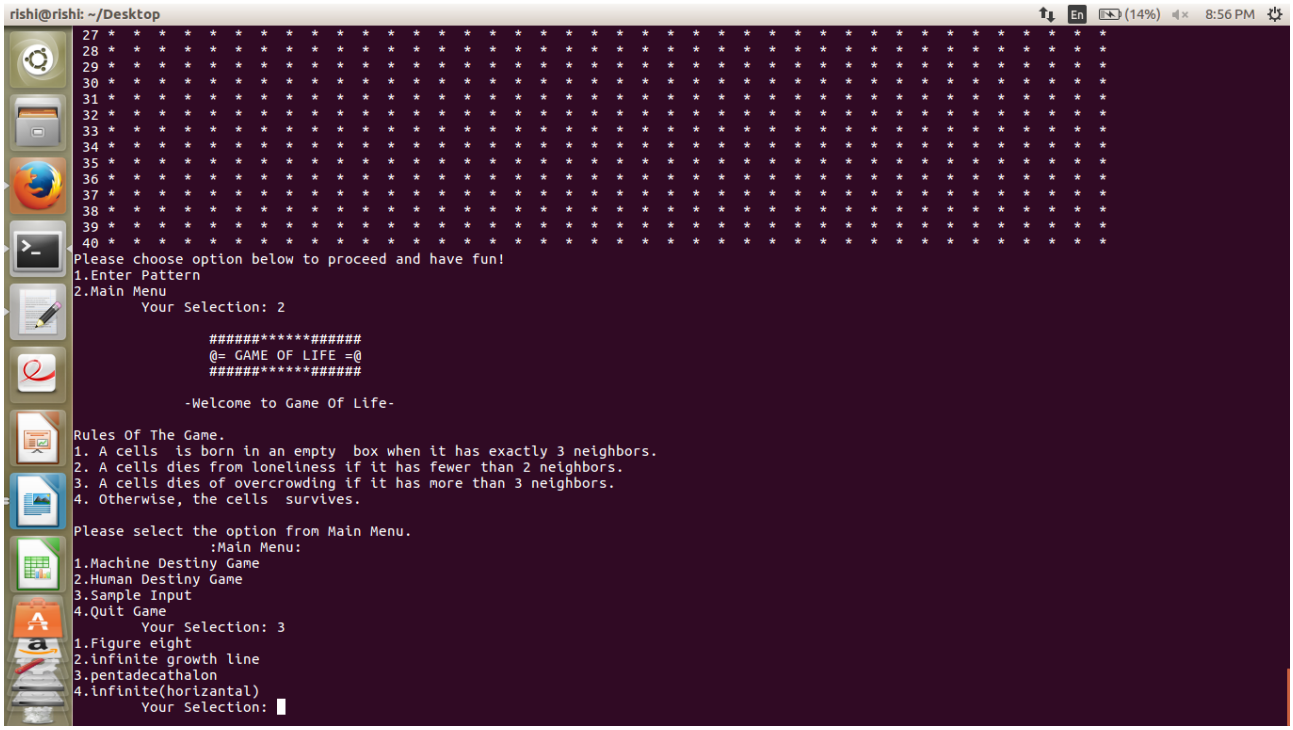

rishi@rishi: ~/Desktop
En
(13%)
8:53 PM
@Machine Destiny Game@

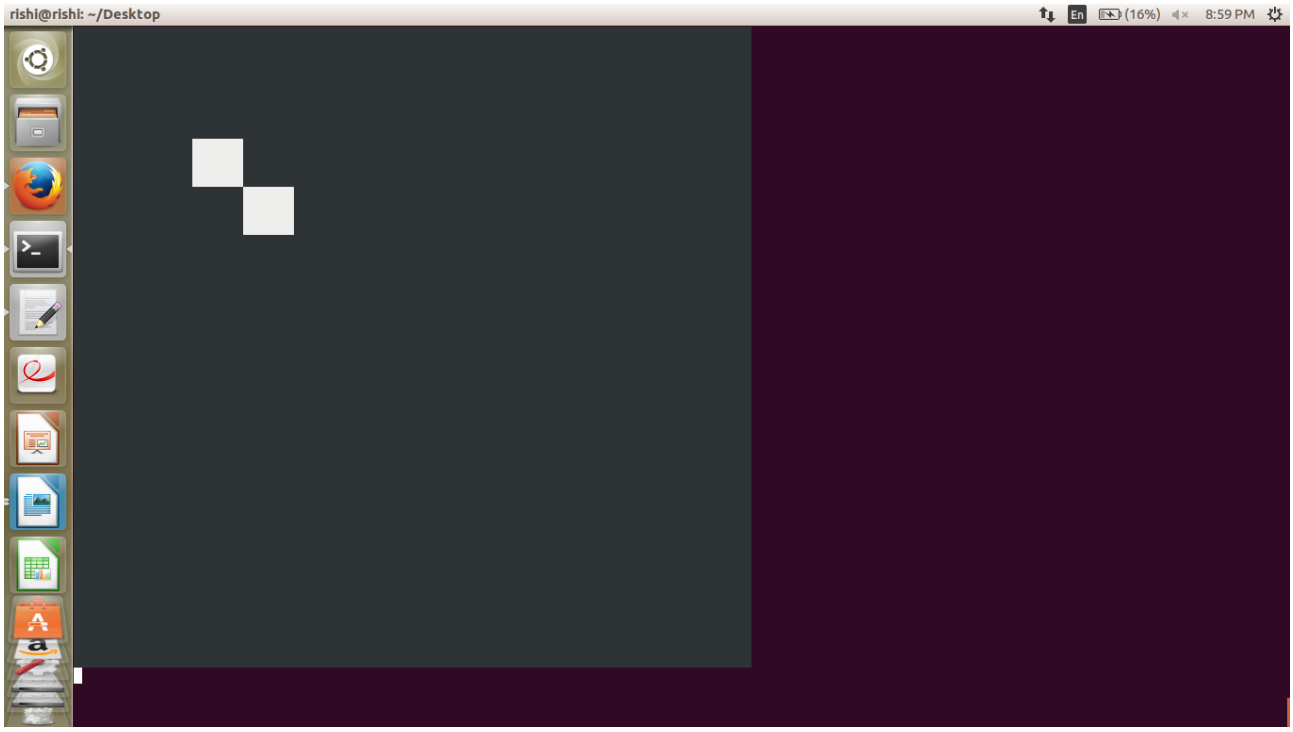

rishi@rishi: ~/Desktop
En
(13%)
8:54 PM
@Machine Destiny Game@

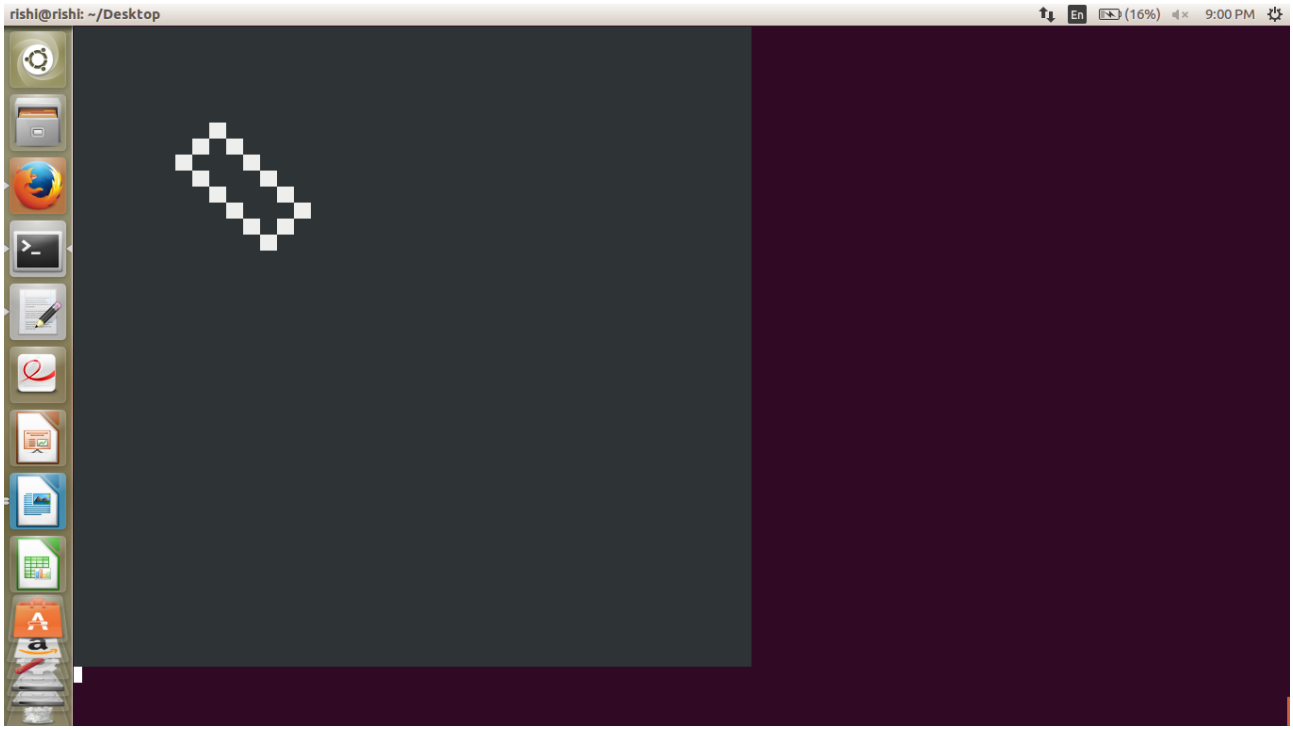

rishi@rishi: ~/Desktop
En
(13%)
8:54 PM
@Machine Destiny Game@

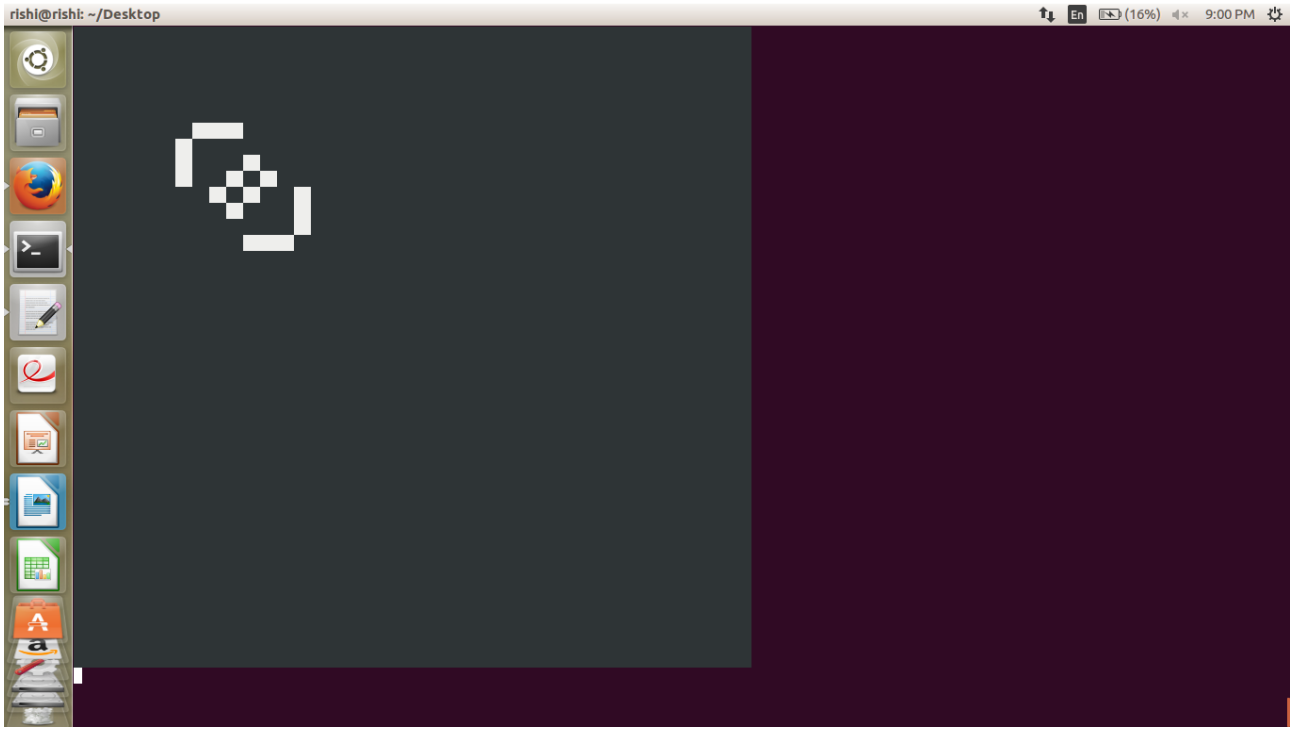

rishi@rishi: ~/Desktop
(14%) 8:55 PM
############
@= GAME OF LIFE =@
############
-Welcome to Game Of Life-
Rules Of The Game.
1. A cells is born in an empty box when it has exactly 3 neighbors.
2. A cells dies from loneliness if it has fewer than 2 neighbors.
3. A cells dies of overcrowding if it has more than 3 neighbors.
4. Otherwise, the cells survives.
Please select the option from Main Menu.
:Main Menu:
1.Machine Destiny Game
2.Human Destiny Game
3.Sample Input
4.Quit Game
Your Selection: 2
@ Human Destiny Game @
In this game, you are free to choose the pattern of cell.
Then, you can see the cell progress based on your cell pattern.
1.Want to see the scale of the plane
2.Continue with the game
enter your choice :

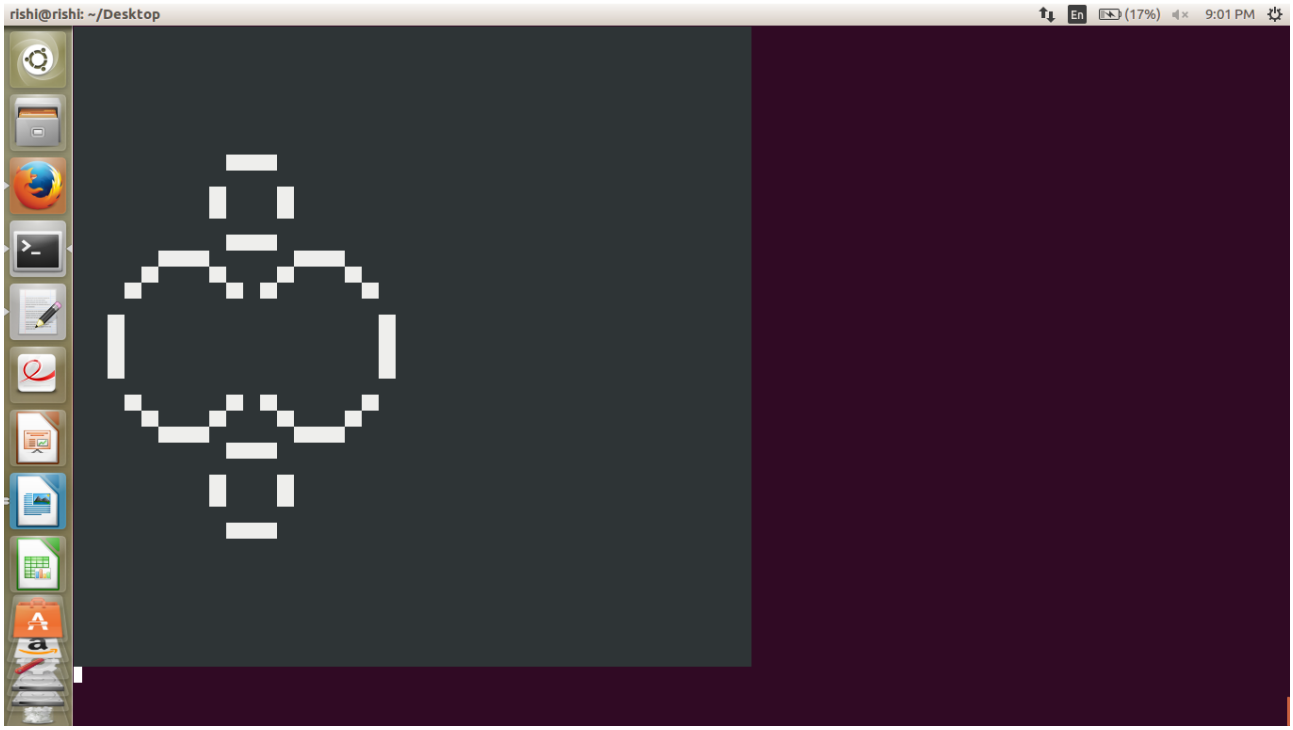

rishi@rishi: ~/Desktop
(14%) 8:56 PM
############
@= GAME OF LIFE =@
############
-Welcome to Game Of Life-
Rules Of The Game.
1. A cells is born in an empty box when it has exactly 3 neighbors.
2. A cells dies from loneliness if it has fewer than 2 neighbors.
3. A cells dies of overcrowding if it has more than 3 neighbors.
4. Otherwise, the cells survives.
Please select the option from Main Menu.
:Main Menu:
1.Machine Destiny Game
2.Human Destiny Game
3.Sample Input
4.Quit Game
Your Selection: 2
@ Human Destiny Game @
In this game, you are free to choose the pattern of cell.
Then, you can see the cell progress based on your cell pattern.
1.Want to see the scale of the plane
2.Continue with the game
enter your choice : 1
0 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
01 02 03 04 05 06 07 08 09 10 11 12 13 14


rishi@rishi: ~/Desktop
En
(14%)
8:56 PM
Please choose option below to proceed and have fun!
1.Enter Pattern
2.Main Menu
Your Selection: 2
@= GAME OF LIFE =@
-Welcome to Game Of Life-
Rules Of The Game.
1. A cells is born in an empty box when it has exactly 3 neighbors.
2. A cells dies from loneliness if it has fewer than 2 neighbors.
3. A cells dies of overcrowding if it has more than 3 neighbors.
4. Otherwise, the cells survives.
Please select the option from Main Menu.
:Main Menu:
1.Machine Destiny Game
2.Human Destiny Game
3.Sample Input
4.Quit Game
Your Selection: 3
1.Figure eight
2.infinite growth line
3.pentadecathalon
4.infinite(horizantal)
Your Selection:

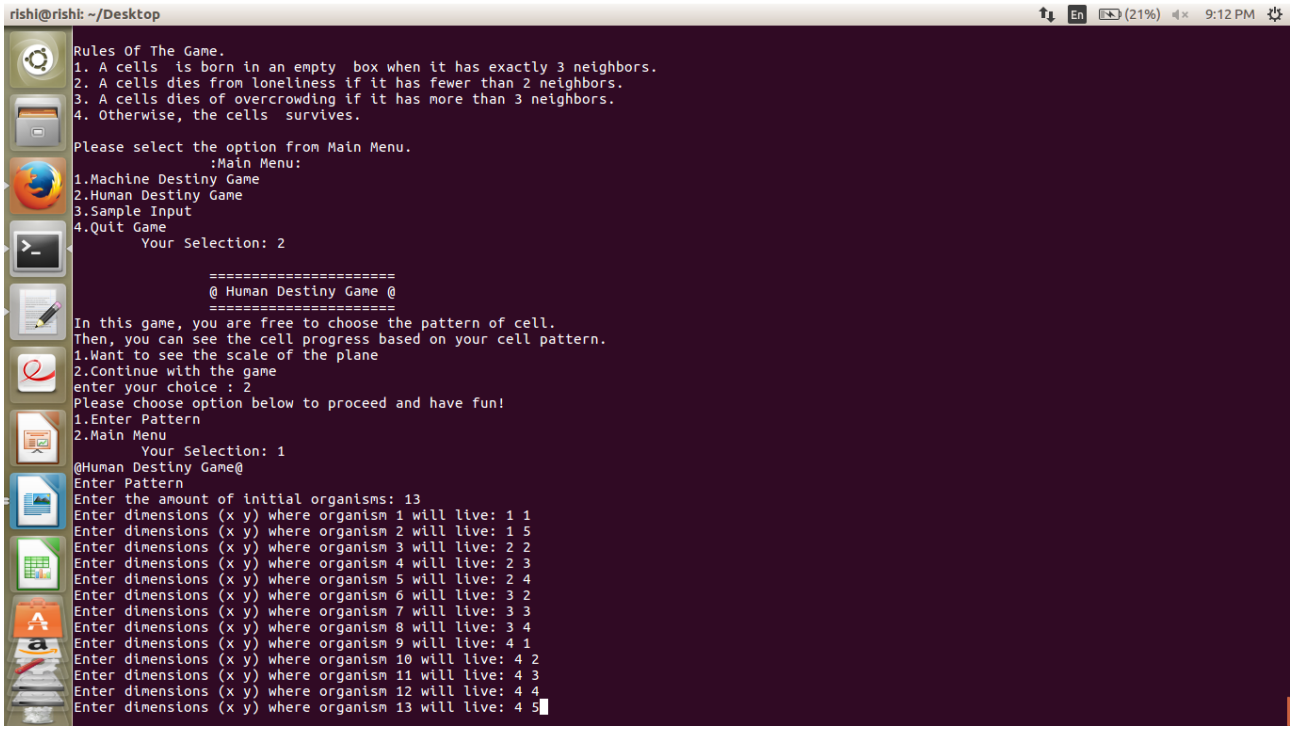

rishi@rishi: ~/Desktop
En
(16%)
8:59 PM

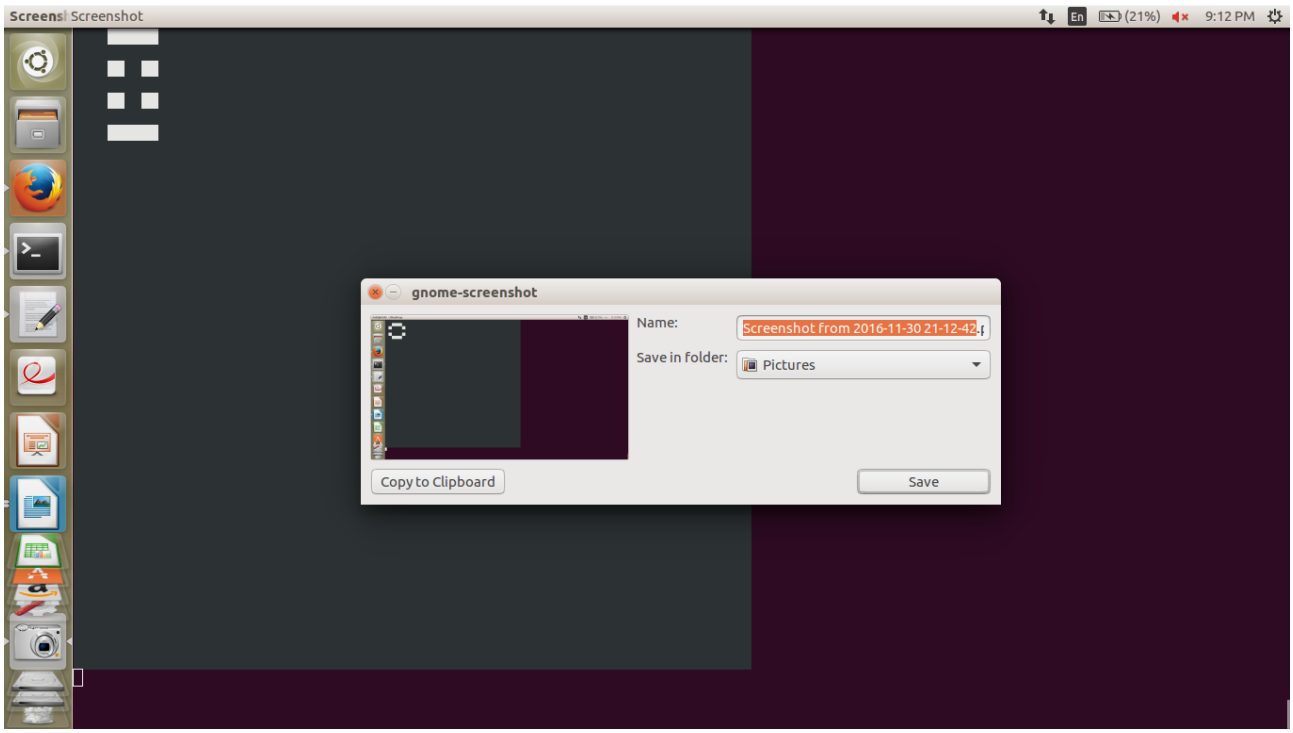

rishi@rishi: ~/Desktop
En
(16%)
9:00 PM

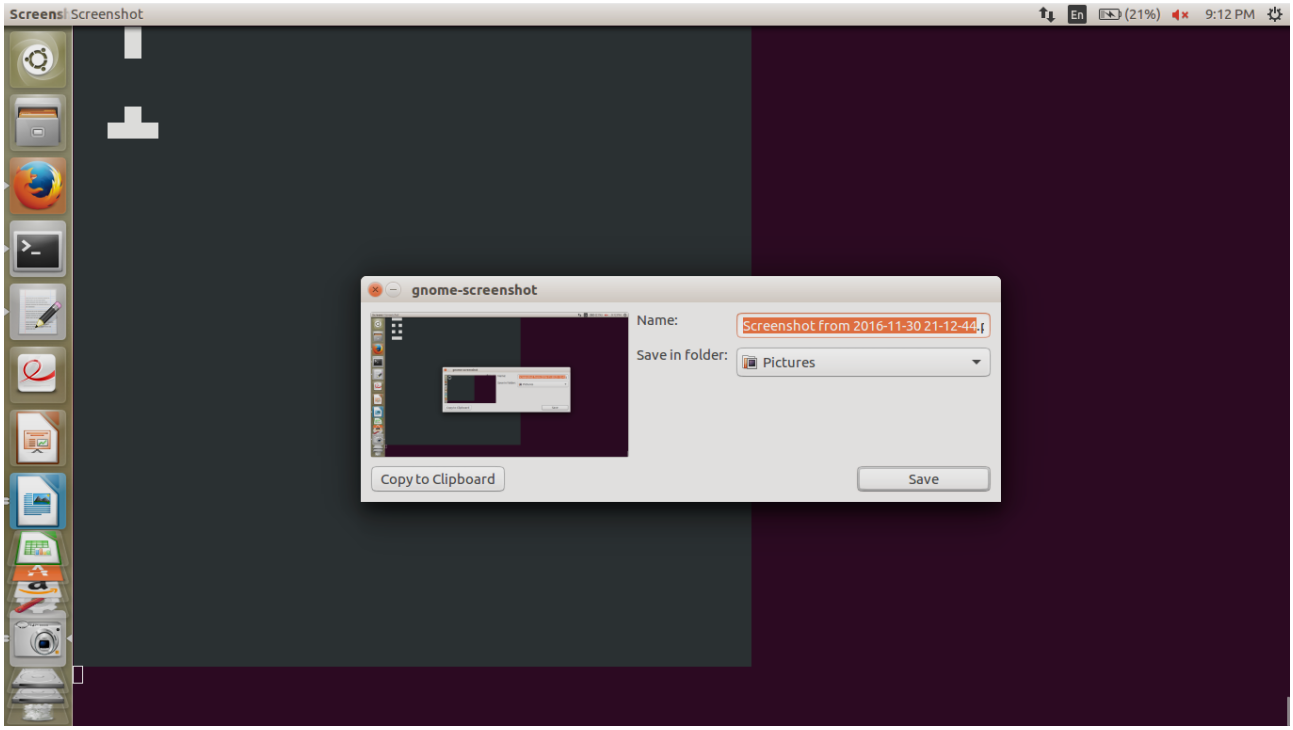

rishi@rishi: ~/Desktop
En
(16%)
9:00 PM


rishi@rishi: ~/Desktop
En
(17%)
9:01 PM

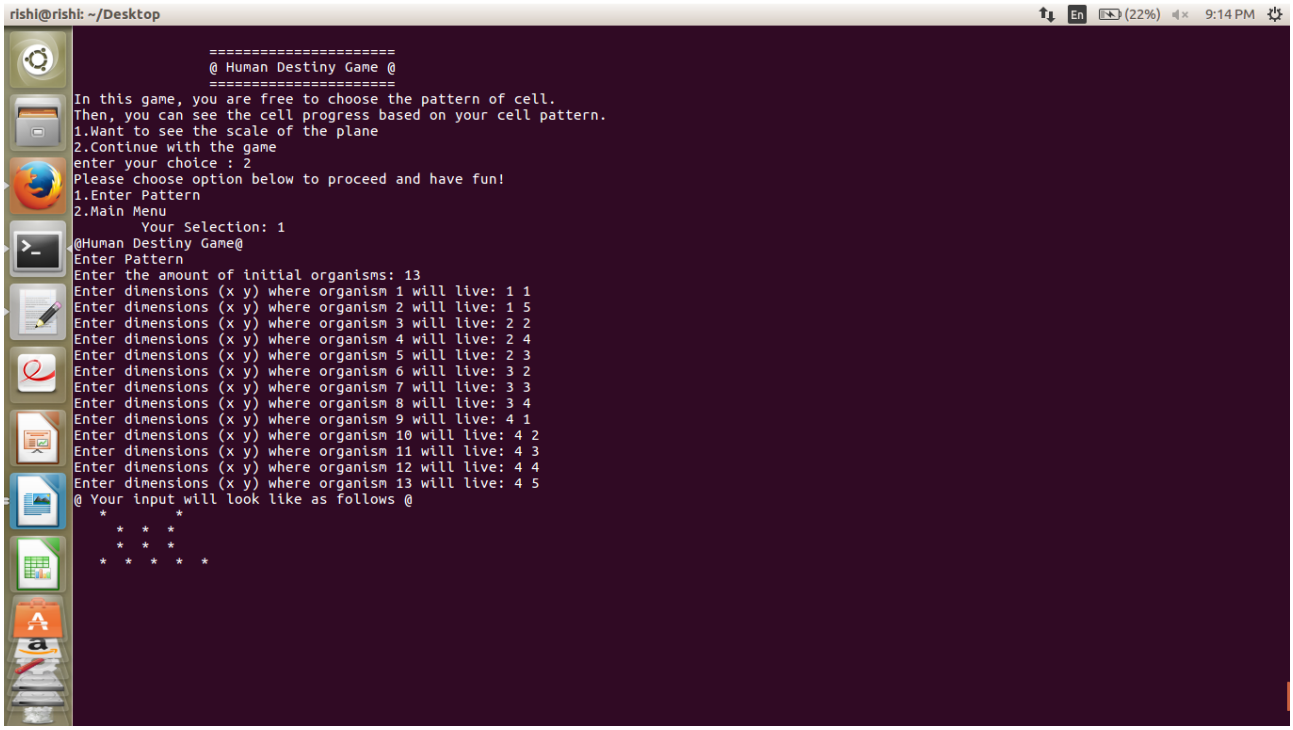

rishi@rishi: ~/Desktop
En
(17%)
9:01 PM


rishi@rishi: ~/Desktop
En (21%) 9:12 PM
Rules Of The Game.
1. A cells is born in an empty box when it has exactly 3 neighbors.
2. A cells dies from loneliness if it has fewer than 2 neighbors.
3. A cells dies of overcrowding if it has more than 3 neighbors.
4. Otherwise, the cells survives.
Please select the option from Main Menu.
:Main Menu:
1.Machine Destiny Game
2.Human Destiny Game
3.Sample Input
4.Quit Game
Your Selection: 2
==========
@ Human Destiny Game @
==========
In this game, you are free to choose the pattern of cell.
Then, you can see the cell progress based on your cell pattern.
1.Want to see the scale of the plane
2.Continue with the game
enter your choice : 2
Please choose option below to proceed and have fun!
1.Enter Pattern
2.Main Menu
Your Selection: 1
@Human Destiny Game@
Enter Pattern
Enter the amount of initial organisms: 13
Enter dimensions (x y) where organism 1 will live: 1 1
Enter dimensions (x y) where organism 2 will live: 1 5
Enter dimensions (x y) where organism 3 will live: 2 2
Enter dimensions (x y) where organism 4 will live: 2 3
Enter dimensions (x y) where organism 5 will live: 2 4
Enter dimensions (x y) where organism 6 will live: 3 2
Enter dimensions (x y) where organism 7 will live: 3 3
Enter dimensions (x y) where organism 8 will live: 3 4
Enter dimensions (x y) where organism 9 will live: 4 1
Enter dimensions (x y) where organism 10 will live: 4 2
Enter dimensions (x y) where organism 11 will live: 4 3
Enter dimensions (x y) where organism 12 will live: 4 4
Enter dimensions (x y) where organism 13 will live: 4 5


Screens Screenshot
En (21%) 9:12 PM
gnome-screenshot
Name:
Screenshot from 2016-11-30 21-12-42.
Save in folder:
Pictures
Copy to Clipboard
Save


Screens| Screenshot
En (21%) 9:12 PM
gnome-screenshot
Name:
Screenshot from 2016-11-30 21-12-44.
Save in folder:
Pictures
Copy to Clipboard
Save


rishi@rishi: ~/Desktop
En (22%) 9:14 PM



FUTURE IMPROVEMENTS:

> we can make it colourfull by giving differrent colours to live cell living in next generation and other dying in next generation.

> we can use graphics and make it look better

>we can ask the user for the time delay like the printing should be slow or medium or fast

SOURCES USED:

for echo commands:

stack overflow

git hub

wikipedia

for random function:

tutorial point

for fflush(stdout):

stack overflow

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

#define HEIGHT 40
#define WIDTH 40
#define LIVE_CELL 1
#define DEAD_CELL 0

typedef int TableType[HEIGHT][WIDTH];

void srand(unsigned int seed);                  //pseudo random number
generator
int main_menu();                        //main menu function
int machine_game_menu();                //machine menu function
int human_game_menu();                  //human menu function
void end_game();                        //end game function
void load_random(TableType table);      //generate random pattern
void user_input(TableType table);       //generate user pattern

int neighbour_value(TableType table, int row, int col); // value of
neighbour
int neighbour_count(TableType table, int row, int col); // counts the
number of neighbour
void calculate(TableType table);        // calculates according to the rules
void print_user_pattern(TableType table);  // prints the generations of
user
void print_machine_pattern(TableType table);    // print the generations
of machine
int sample_menu();                      // menu of different patterns
void sample_input(TableType table);     // prints the some beautiful
outcomes

void print_user_pattern(TableType table)
{
        int row, col;

//      ======================
//      @ Human Destiny Game @
```

```
//      ======================

        printf("\033[H");
        for ( row = 0; row < HEIGHT; row++ )
        {
                for ( col = 0; col < WIDTH; col++ )
                {
                        if (table[row][col] == LIVE_CELL)
                        {
                                printf("\033[07m  \033[0m\033[37m\033[40;01m");
                        }
                        else
                        {
                                printf("  ");
                        }
                }
                printf("\033[E");
        }
        fflush(stdout);
}

void print_machine_pattern(TableType table)
{
        int row, col;

//      ======================
//      @Machine Destiny Game@
//      ======================

        printf("\033[H");
        for ( row = 0; row < HEIGHT; row++ )
        {
                for ( col = 0; col < WIDTH; col++ )
                {
                        if (table[row][col] == LIVE_CELL)
                        {
                                printf("\033[07m  \033[0m\033[37m\033[40;01m");
                        }
                        else
                        {
                                printf("  ");
                        }
```

```
            }
            printf("\033[E");
        }
        fflush(stdout);
}

void user_input(TableType table)
{
        int y,z;
        for(y=0;y<HEIGHT;y++)
        {
                for(z=0;z<WIDTH;z++)
                {
                        table[y][z]=DEAD_CELL;
                }
        }

        int i,j;
        int n;
        int height, width;


        printf("Enter the amount of initial organisms: ");
        scanf("%d", &n);
        for (i = 0; i < n; i++)
        {
                printf("Enter dimensions (x y) where organism %d will live: ", i
+ 1);
                scanf("%d %d", &height, &width);
                table[height][width] = LIVE_CELL;
        }
        printf("@ Your input will look like as follows @");
        for(i = 0; i < HEIGHT; i++)
        {
                for(j = 0; j < WIDTH; j++)
                {
                        if( table[i][j] == LIVE_CELL )
                                {printf(" * ");}
                        else
                                printf("  ");
                }
                printf("\n");
```

```
        }
        int w=0;
        while(w<20)
        {
                printf("\n");
        w++;
        }
}

void load_random(TableType table)
{
        int y,z;
        for(y=0;y<HEIGHT;y++)
        {
                for(z=0;z<WIDTH;z++)
                {
                        table[y][z]=DEAD_CELL;
                }
        }

        int x, row, col, i;
        x = ( rand() % 1000 ) + 10;
        for(i = 1;i <= x; i++)
        {
                row = ( rand() % 60) + 4;
                col = ( rand() % 60 ) + 4;
                table[row][col] = LIVE_CELL;
        }
}

int neighbour_value(TableType table, int row, int col)
{
        if (row < 0 || row >= HEIGHT
                || col < 0 || col >= WIDTH
                || table[row][col] != LIVE_CELL )
        {
                return 0;
        }
        else
        {
                return 1;
        }
```

```
}

int neighbour_count(TableType table, int row, int col)
{
	int neighbour = 0;

	neighbour += neighbour_value(table, row - 1, col - 1);
	neighbour += neighbour_value(table, row - 1, col);
	neighbour += neighbour_value(table, row - 1, col + 1);
	neighbour += neighbour_value(table, row, col - 1);
	neighbour += neighbour_value(table, row, col + 1);
	neighbour += neighbour_value(table, row + 1, col - 1);
	neighbour += neighbour_value(table, row + 1, col);
	neighbour += neighbour_value(table, row + 1, col + 1);

	return neighbour;
}

void calculate(TableType table)
{
	TableType tableB;
	int neighbour, height, width;

	for (height = 0; height < HEIGHT; height++)
	{
		for (width = 0; width < WIDTH; width++)
		{
			neighbour = neighbour_count(table, height, width);
			if (neighbour==3)
			{
				tableB[height][width] = LIVE_CELL;
			}
			else if (neighbour == 2 && table[height][width] ==
LIVE_CELL)
			{
				tableB[height][width] = LIVE_CELL;
			}
			else
			{
				tableB[height][width] = DEAD_CELL;
			}
		}
```

```
        }
        for (height = 0; height < HEIGHT; height++)
        {
                for (width = 0; width < WIDTH; width++)
                {
                        table[height][width] = tableB[height][width];
                }
        }
}

int main_menu()
{
    int opt1;
    printf("\n\t\t######******######\n");
        printf("\t\t@= GAME OF LIFE =@\n");
        printf("\t\t######******######\n");
        printf("\n\t     -Welcome to Game Of Life-\n");
        printf("\nRules Of The Game.\n");
        printf("1. A cells  is born in an empty  box when it has exactly 3
neighbors.\n");
        printf("2. A cells dies from loneliness if it has fewer than 2
neighbors.\n");
        printf("3. A cells dies of overcrowding if it has more than 3
neighbors.\n");
        printf("4. Otherwise, the cells  survives.\n");
        printf("\nPlease select the option from Main Menu.\n");
        printf("\t\t:Main Menu:\n");
        printf("1.Machine Destiny Game\n");
        printf("2.Human Destiny Game\n");
        printf("3.Sample Input\n");
        printf("4.Quit Game\n");
        printf("\tYour Selection: ");
    scanf ("%d", &opt1);
    return opt1;

}

int machine_game_menu()
{
     int opt2;
     printf("\n\t\t=======================\n");
     printf("\t\t@ Machine Destiny Game @\n");
```

```
	printf("\t\t=======================\n");
	printf("In this game, computer will generate the pattern of cell
randomly for you.\n");
	printf("All you have to do is sit back and watch the cell progress.\n");
	printf("Please choose option below to proceed and have fun!\n");
	printf("1.Start Game\n");
	printf("2.Main Menu\n");
	printf("\tYour Selection: ");
	scanf ("%d", &opt2);
	int i=0;
	  while(i<50)
	  {
		printf("\n");
	  i++;
	  }
	return opt2;
}

int human_game_menu()
{
    int opt3, i, j, t;
    printf("\n\t\t=======================\n");
    printf("\t\t@ Human Destiny Game @\n");
    printf("\t\t=======================\n");
    printf("In this game, you are free to choose the pattern of cell.\n");
    printf("Then, you can see the cell progress based on your cell
pattern.\n");
    printf("1.Want to see the scale of the plane\n2.Continue with the
game\nenter your choice : ");
    scanf("%d",&t);

	  if(t==1)
	  {
		for(i=0; i<=HEIGHT ;i++)
		{
			for(j=0; j<=WIDTH; j++)
			{
		      if(i==0)
				{
					if(j<10 && j!=0)
					 {
				    printf(" 0%d",j);
```

```
                        }
                    else
                        {
                        printf(" %d",j);
                        }
                }
            else
                {
                    if(j==0)
                    {
                       if(i<10)
                        {
                           printf(" 0%d",i);
                        }
                       else
                        {
                            printf(" %d",i);
                        }
                    }
                    else
                        printf(" * ");
                }
            }
            printf("\n");
        }
    }
    printf("Please choose option below to proceed and have fun!\n");
    printf("1.Enter Pattern\n");
    printf("2.Main Menu\n");
    printf("\tYour Selection: ");
    scanf ("%d", &opt3);

    return opt3;
}

void end_game()
{
    printf("\n\t\t============\n");
    printf("\t\t@ Quit Game @\n");
    printf("\t\t============\n");
    printf("Thank you for trying this game. I hope you have found a lot
of fun here.\n");
```

```
}


int sample_menu()
{
	printf("1.Figure eight\n2.infinite growth line\n3.Tumbler\
n4.infinite(horizantal)\n5.R-pentimino\n6.DieHardn");
	int i;
	printf("\tChoose number so that you could observe beautiful
patterns: ");
	scanf("%d",&i);
return i;
}




void sample_input(TableType table)
{
	int t,i,j;
	for(i=0;i<HEIGHT;i++)
	{
		for(j=0;j<WIDTH;j++)
		{
			table[i][j]=0;
		}
	}

	t= sample_menu();
	if(t==1)
	{
		for(i=7;i<10;i++)
		{
			for(j=7;j<10;j++)
			{
			    table[i][j]=LIVE_CELL;
			}
		}
		for(i=10;i<13;i++)
		{
			for(j=10;j<13;j++)
			{
```

```
                table[i][j]=LIVE_CELL;
            }
        }
}
if(t==2)
{
        for(j=10;j<30;j++)
        {
          table[j][10]=LIVE_CELL;
        }
}
if(t==3)
{

        table[1][2]=LIVE_CELL;
        table[2][1]=LIVE_CELL;
        table[3][1]=LIVE_CELL;
        table[2][3]=LIVE_CELL;
        table[3][4]=LIVE_CELL;
        table[4][3]=LIVE_CELL;
        table[5][3]=LIVE_CELL;
        table[5][4]=LIVE_CELL;
        table[5][6]=LIVE_CELL;
        table[5][7]=LIVE_CELL;
        table[4][7]=LIVE_CELL;
        table[3][6]=LIVE_CELL;
        table[3][9]=LIVE_CELL;
        table[2][7]=LIVE_CELL;
        table[2][9]=LIVE_CELL;
        table[1][8]=LIVE_CELL;

}
if(t==4)
  {     int j;

        for(j=10;j<30;j++)
        {
          table[10][j]=LIVE_CELL;
        }
}
if(t==5)
{
```

```
            table[14][10]=LIVE_CELL;
            table[14][11]=LIVE_CELL;
            table[15][9]=LIVE_CELL;
            table[15][10]=LIVE_CELL;
            table[16][10]=LIVE_CELL;
        }
        if(t==6)
        {
            table[15][5]=LIVE_CELL;
            table[15][6]=LIVE_CELL;
            table[16][6]=LIVE_CELL;
            table[16][10]=LIVE_CELL;
            table[14][11]=LIVE_CELL;
            table[16][11]=LIVE_CELL;
            table[16][12]=LIVE_CELL;
        }
        int u=0;
        while(u != 100)
        {
                printf("\n");
         u++;
        }
}



int main()
{
  int opt1, opt2, opt3, opt4, k = 1;

  TableType table;
  srand ( time ( NULL ) );


        do
        {
            opt1 = main_menu();
            if (opt1 == 1)
            {
                do
                {
                   opt2 = machine_game_menu();
```

```
                if(opt2 == 1)
                {
                   printf("@Machine Destiny Game@\n");
                    load_random( table );
                    do
                    {
                        print_machine_pattern( table );
                        calculate( table );
                        usleep(450000);
                    }while(1);
                }
                if(opt2 > 2)
                {
                        printf("\t\t-WRONG INPUT-\n\n");
                }
        }while(opt2 > 2);
}
else if (opt1 == 2)
{
   do
   {
         opt3 = human_game_menu();
                if(opt3 == 1)
                {
                    printf("@Human Destiny Game@\n");
                    printf("Enter Pattern\n");
                    user_input( table );
                    do
                    {
                        print_user_pattern( table );
                        calculate( table );
                        usleep(900000);
                    }while(1);
                }
                if(opt3 > 2)
                {
                        printf("\t\t-WRONG INPUT-\n\n");
                }
    }while(opt3 > 2);
}
else if(opt1 == 3)
{
```

```
            sample_input(table);
                        do
                           {
                               print_user_pattern( table );
                               calculate( table );
                               usleep(450000);
                           }while(1);


           }

            else if (opt1 == 4)
            {
                end_game();
                break;
            }
            else
            {
                printf("\t\t-WRONG INPUT-\n\n");
            }
        }while( opt1 > 3 || opt2 == 2 || opt3 == 2 || opt4 == 2 );



  return 0;
}
```