**SUDOKU GAME**

#include<iostream>

#include<cstdlib>

#include<ctime>

#include<cmath>

#include<conio.h>

#define cps 1000;

using namespace std;

class Sudoku{

public:

int mat[9][9];

int K; // No. Of missing digits

// Constructor

Sudoku( int k){

K = k;

}

// Sudoku Generator

void fillValues(){

// Fill the diagonal of 3 x 3 matrices

fillDiagonal();

// Fill remaining blocks

fillRemaining(0, 3);

// Remove Randomly K digits to make game

removeKDigits();

}

// Fill the diagonal SRN number of SRN x SRN matrices

void fillDiagonal(){

for (int i = 0; i<9; i=i+3)

// for diagonal box, start coordinates->i==j

fillBox(i, i);

}

// Returns false if given 3 x 3 block contains num.

bool unUsedInBox(int rowStart, int colStart, int num){

for (int i = 0; i<3; i++)

for (int j = 0; j<3; j++)

if (mat[rowStart+i][colStart+j]==num)

return false;

return true;

}

// Fill a 3 x 3 matrix.

void fillBox(int row,int col){

int num;

for (int i=0; i<3; i++){

for (int j=0; j<3; j++){

do{

num = randomGenerator(9);

}

while (!unUsedInBox(row, col, num));

mat[row+i][col+j] = num;

}

}

}

// Random generator

int randomGenerator(int num){

int r=(rand()%num)+1;

return r;

}

// Check if safe to put in cell

bool CheckIfSafe(int i,int j,int num){

return (unUsedInRow(i, num) && unUsedInCol(j, num) && unUsedInBox(i-i%3, j-j%3, num));

}

// check in the row for existence

bool unUsedInRow(int i,int num){

for (int j = 0; j<9; j++)

if (mat[i][j] == num)

return false;

return true;

}

// check in the row for existence

bool unUsedInCol(int j,int num){

for (int i = 0; i<9; i++)

if (mat[i][j] == num)

return false;

return true;

}

// A recursive function to fill remaining matrix

bool fillRemaining(int i, int j){

if (j>=9 && i<8){

i = i + 1;

j = 0;

}

if (i>=9 && j>=9)

return true;

if (i < 3){

if (j < 3)

j = 3;

}

else if (i < 6){

if (j==(int)(i/3)\*3)

j = j + 3;

}

else{

if (j == 6){

i = i + 1;

j = 0;

if (i>=9)

return true;

}

}

for (int num = 1; num<=9; num++){

if (CheckIfSafe(i, j, num)){

mat[i][j] = num;

if (fillRemaining(i, j+1))

return true;

mat[i][j] = 0;

}

}

return false;

}

// Remove the K no. of digits to complete game

void removeKDigits(){

int cnt = K;

while (cnt != 0){

int cellId = randomGenerator(81);

// extract coordinates i and j

int i = (cellId/9);

int j = cellId%9;

if (j != 0)

j = j - 1;

if (mat[i][j] != 0){

cnt--;

mat[i][j] = 0;

}

}

}

// Print sudoku

void printSudoku(){

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

for (int j = 0; j<9; j++)

cout<<mat[i][j]<<" ";

cout<<endl;

}

cout<<endl;

}

};

//To find whether a number can be placed at a position or not

bool canPlace(int in[9][9], int i, int j, int no) {

int x, y;

//If number is already present in the row return false

for (x = 0; x < 9; x++) {

if (in[i][x] == no or in[x][j] == no)

return false;

}

//to check if the number is present in the submatrix or not

x = (i / 3) \* 3;

y = (j / 3) \* 3;

for (int a=x; a < x + 3; a++) {

for (int b=y; b < y + 3; b++) {

if (in[a][b] == no)

return false;

}

}

//if above conditions are satisfied return true

return true;

}

bool solve(int in[9][9], int i, int j) {

//Base case to print matrix

if (i == 9) {

return true;

}

//Move to next row

if (j == 9) {

return solve(in, i + 1, 0);

}

//Already filled

if (in[i][j] != 0) {

return solve(in, i, j + 1);

}

//Check what number to place in this position

for (int no = 1; no <= 9; no++) {

if (canPlace(in, i, j, no)) {

in[i][j] = no;

bool a = solve(in, i, j + 1);

if (a==true) {

return true;

}

}

}

//if input matrix is incorrect

in[i][j] = 0;

return false;

}

//parameters for the game

int count=81,lives=100,score=1000;

// timers for storing current and previous time

clock\_t time\_curr=(float)clock()/cps;

clock\_t time\_prev=(float)clock()/cps;

void updateScore(){

if(time\_curr-time\_prev>=2) score=score-(time\_curr-time\_prev)/2;

}

//to print sudoku with formatting

void output(int mat[9][9]){

system("cls");

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

cout<<"--";

}

cout<<"-"<<endl;

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

cout<<"|";

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

if((j+1)%3!=0)

cout<<mat[i][j]<<" ";

else

cout<<mat[i][j]<<"|";

}

cout<<endl;

if((i+1)%3==0){

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

cout<<"--";

}

cout<<"-"<<endl;

}

}

}

// func. to get running input

void input(int mat[9][9],int mat2[9][9] ){

cout<<"enter row and coloumn no."<<endl;

int i,j;

cin>>i>>j;

i--;

j--;

if(mat[i][j]!=0){

cout<<"position incorrect\nPress enter to continue"<<endl;

getch();

cin.sync();

}

else{

cout<<"enter value to be entered "<<endl;

int val;

cin>>val;

if(mat2[i][j]==val){

mat[i][j]=val;

count--;

}

else{

cout<<"incorrect value\nPress enter to continue"<<endl;

lives--;

getch();

cin.sync();

}

}

output(mat);

}

// Driver code

int main(){

srand(time(0));

//Number of zeroes in Sudoku

int K;

int x;

lives=10;

a:

cout<< "Enter difficulty level\n1.Hard 2.Medium 3.Easy"<<endl;

cin>>x;

//Choose difficulty

switch(x){

case 1: {K=50;lives =20; break;}

case 2: K=40; break;

case 3: K=30; break;

default:

{

cout<<"enter valid difficulty level";

goto a;

}

}

count=K;

//Create sudoku with zeroes

Sudoku\* sudoku = new Sudoku( K);

sudoku->fillValues();

int matrix[9][9];

int solution[9][9];

//Copying sudoku into 2 matrices

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

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

matrix[i][j]=(sudoku->mat)[i][j];

solution[i][j]=(sudoku->mat)[i][j];

}

}

// solution matrix

solve(solution,0,0);

output(matrix);

time\_curr=(float)clock()/cps;

// continue game until lives are empty or score is 0

while(lives && score){

input(matrix,solution);

time\_prev=time\_curr;

time\_curr=(float)clock()/cps;

updateScore();

//if sudoku is complete

if(count==0){

cout<<"Congratulations!!! You solved the Sudoku successfully with score of "<<score;

break;

}

else

cout<<"You have "<<lives<<" lives left and Your current score is: "<<score<<endl;

cout<<"Time consumed is: "<<time\_curr/60<< " min "<<time\_curr%60<<" sec"<<endl;

}

// if users ends up losing the game

if(lives==0 || score==0)

{

cout<<"Sorry! You lost the game. Give another try. press enter for the solution"<<endl;

getch();

output(solution);

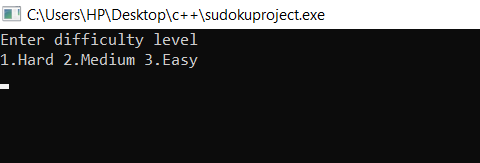
cout<<"This was the solution to given sudoku";

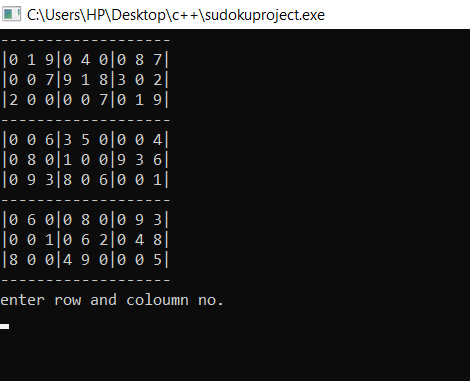
}

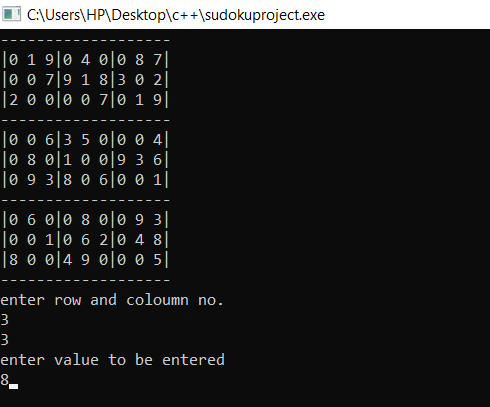
return 0;

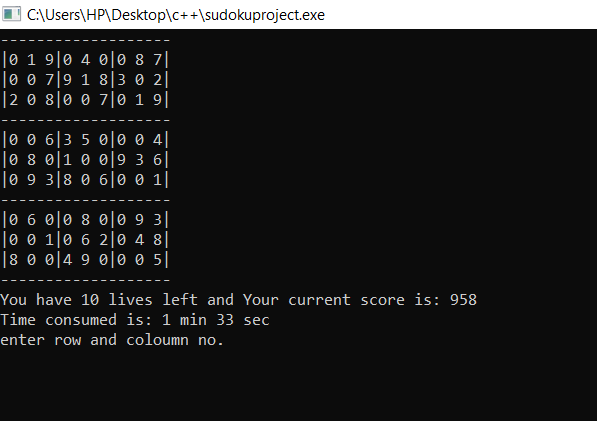
}

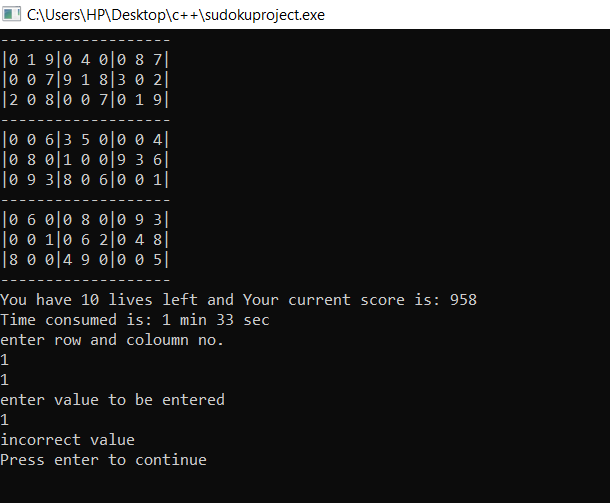
OUTPUT:-

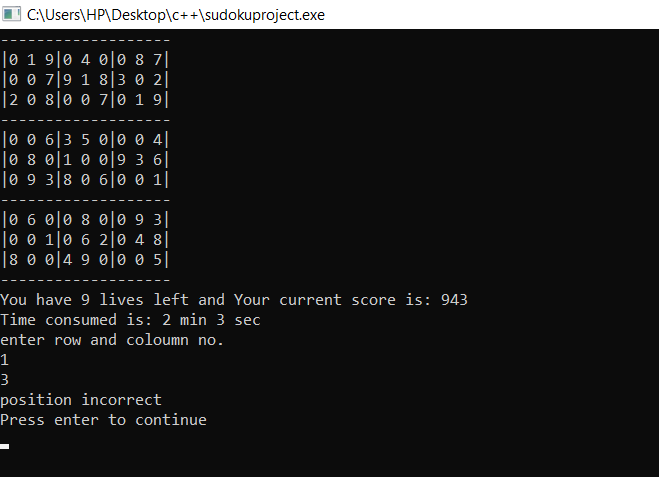












\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_