**PROGRAM CODE:**

#include<ctype.h>

#include<graphics.h>

#include<iostream.h>

#include<stdlib.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<string.h>

#include<process.h>

#include<dos.h>

#include<math.h>

#include<time.h>

#include<stdlib.h>

// Bitmap image structures :

struct bmpfileheader {

char type[2]; // Here are the letters { 'B' , 'M '}

long int size; //Size of the entire file

char reserved[4];

long int offbits; //Offset of the bitmap in the file

};

struct bmpinfoheader {

long int size; //Size of bmpinfoheader

//12 - OS/2 1.x format

//40 - Windows 3.x format

//64 - OS/2 2.x format

long int width; //Horizontal width of bitmap in pixels

long int height; //Vertical height of bitmap in pixels

int planes; //Number of planes

int bitcount; //bits per pixel: 1, 4, 8, 24

//This then indicates the number of colors

long int compression; //0 - none

//1 - RLE - 8 bit/pixel

//2 - RLE - 4 bit/pixel

long int sizeimage; //Size of the image in bytes

long int xpm; //Horizontal Resolution: x pixels per meter

long int ypm; //Vertical Resolution: y pixels per meter

long int colused; //Number of colors used

long int colimp; //Number of important colors

};

struct RGBquad

{

unsigned char blue,green,red;

unsigned char resv; //reserved

};

//void bmp\_save(int x1,int y1,int x2,int y2,char \*filename);

RGBquad \*bmp\_load(int x1,int y1,char \*filename);

int init\_graph();

void bmp\_save(int x1,int y1,int x2,int y2,char \*filename);

//ONE MORE LINE

int z,r1,score=0,x2,y2;

double x1,y1,b=0;

static double u=0.016;

int w=3;

//GIVING COORDINATES IN TERMS OF GETMAXX AND GETMAXY AND Z ETC FOR THE GAME

//TO WORK IN ALL RESOLUTIONS

void background()

{//F1

//DRAWING THE 4 BACKGROUND LINES

if(getmaxx()<getmaxy())

{

setcolor(w);

line(0,0,getmaxx(),0);

line(0,0,0,getmaxx());

line(0,getmaxx(),getmaxx(),getmaxx());

line(getmaxx(),0,getmaxx(),getmaxx());

// THE 4 CIRCLES

circle(x1,0,r1);

circle(x1,z/2,r1);

circle(x1,z/4,r1);

circle(x1,3\*z/4,r1);

circle(x1,z,r1);

//TIME FUNCTION AND SCORE

cout<<"TIME"<<'\t'<<clock()/CLK\_TCK<<endl;

cout<<"SCORE"<<'\t'<<score;

}//IF

else

{

setcolor(w);

line(0,0,getmaxy(),0);

line(0,0,0,getmaxy());

line(0,getmaxy(),getmaxy(),getmaxy());

line(getmaxy(),0,getmaxy(),getmaxy());

// THE 4 CIRCLES

circle(x1,0,r1);

circle(x1,z/2,r1);

circle(x1,z/4,r1);

circle(x1,3\*z/4,r1);

circle(x1,z,r1);

//TIME FUNCTION AND SCORE

cout<<"TIME"<<'\t'<<clock()/CLK\_TCK<<endl;

cout<<"SCORE"<<'\t'<<score;

}//ELSE

return;

}//FUNCTION END(F1)

void onemoreline()

{

clrscr();

//INITIALISING GRAPHICS DRIVER

int gdriver=DETECT,gmode;

initgraph(&gdriver,&gmode,"C:/TURBOC3/BGI");

//CREATING A SQUARE BOARD

if(getmaxx()<getmaxy())

::z=getmaxx();

else

::z=getmaxy();

char c='1';

cout<<"Welcome to One more Line game!\nRules:\nThe ball rotates around each circle.\nPress any key at the correct time so that the ball reaches the next circle\nThe speed of rotation keeps on increasing.\nIf you press the key at the wrong time and if the ball does not reach the next circle, the game ends.\n Press any key to continue..";

//CREATING 3 CASES FOR THE THREE CIRCLES

batman1:

switch(c)

{

case '1'://MIDDLE CIRCLE

{

getch();//CLEARING THE INPUT BUFFER FOR NEXT TURN FOR USING KBHIT

x1=z/2,y1=z/2,r1=20;

while(w!=1)

{

w--;//w VARIABLE FOR RANDOMLY ASSINING COLOURS WHILE TRAVELLING FROM EACH

//CIRCLE

if(w==2)

w=7;

background();

double t1,s;

//INFINITE LOOP TILL GOING TO NEXT CIRCLE OR GAME OVER

while(1)

{

for(double t=0;t<2\*M\_PI;t+=u)

{

x2=r1\*cos(t)+x1;

y2=r1\*sin(t)+y1;

circle(x2,y2,2);

line(x1,y1,x2,y2);

//FOR THE MOVING SMALL CIRCLE IN THE CIRCLE BY USING EQUATION OF CIRCLE

delay(6);

clrscr();

cleardevice();

background();

//MAKING IT GO IN CIRCLE TILL INPUT GOT BY KBHIT FUNCTION

if(kbhit())

{

//FUNCTIONS FOR MAKING THE BALL GO TANGENTIALLY

if(x2<=x1-15 && y2<=y1)

//THE CORRECT POSITION WHICH GOES TO THE NEXT RING

{t1=0;s=-1;

while(x2<=::z+100 || y2<=::z+100)//FOR MAKING IT GO INDEFINATELY TILL GAME

{ //OVER OR NEXT RING

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=z/4)

{

score++;

u+=0.002;//FOR INCREASING SPEED

c='2';//GOING TO THE NEXT CIRCLE BY GOTO FUNCTION AND NEXT CASE

goto batman1;

}

}

}

//ALL THE FOUR REMAINING IFS AND ELSE FOR BALL TO GO TANGENTIALLY AND EXIT

//IF EXIT ENCOUNTERED

else if(x2<=x1 && y2<=y1)

{t1=1;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2>=x1 && y2>=y1)

{t1=-1;s=1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2>=x1 && y2<=y1)

{t1=1;s=1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2<=x1 && y2>=y1)

{t1=-1;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

}

else

continue;//CONTINUING THE LOOP FOR THE BALL TO KEEP MOVING

}

}

}

}//CASE 1

break;

case '2': //CIRCLE 1 UPPER CIRCLE

{

//REMAINING LOGIC SAME AS BEFORE CASE BY ONLY CHANGING COORDINATE OF X1=Z/4

getch();

x1=z/2,y1=z/4,r1=20;

circle(x1,y1,r1);

background();

while(w!=1)

{

w--;

if(w==2)

w=9;

double t1,s;

while(1)

{

for(double t=0;t<2\*M\_PI;t+=u)

{

x2=r1\*cos(t)+x1;

y2=r1\*sin(t)+y1;

circle(x2,y2,2);

line(x1,y1,x2,y2);

delay(6);

clrscr();

cleardevice();

background();

if(kbhit())

{

if(x2<=x1-15 && y2<=y1)

{t1=0;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

score++;

c='3';

u+=0.009;//FOR INCREASING SPEED

goto batman1;

}

}

}

else if(x2<=x1 && y2<=y1)

{t1=1;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2>=x1 && y2>=y1)

{t1=-1;s=1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2>=x1 && y2<=y1)

{t1=1;s=1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2<=x1 && y2>=y1)

{t1=-1;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

}

else

continue;

}

}

}

}//CASE 2

break;

case '3'://CIRCLE 3 LOWER CIRCLE

{

getch();

//JUST CHANGING COORDINATES TO X/2 REMAINING LOGIC SAME

x1=z/2,y1=3\*z/4,r1=20;

circle(x1,y1,r1);

background();

while(w!=1)

{

w--;

if(w==2)

w=9;

double t1,s;

while(1)

{

for(double t=0;t<2\*M\_PI;t+=u)

{

x2=r1\*cos(t)+x1;

y2=r1\*sin(t)+y1;

circle(x2,y2,2);

line(x1,y1,x2,y2);

delay(6);

clrscr();

cleardevice();

background();

if(kbhit())

{

if(x2<=x1-15 && y2<=y1)

{t1=0;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=z/2)

{

score++;

c='1';

u+=0.002;

goto batman1;

}

}

}

else if(x2<=x1 && y2<=y1)

{t1=1;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2>=x1 && y2>=y1)

{t1=-1;s=1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2>=x1 && y2<=y1)

{t1=1;s=1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

if(x2<=x1 && y2>=y1)

{t1=-1;s=-1;

while(x2<=::z+100 || y2<=::z+100)

{

circle(x2+=t1,y2+=s,2);

delay(6);

clrscr();

cleardevice();

background();

if(x2>=::z || y2>=::z || x2<=0 || y2<=0)

{

cout<<endl<<"GAME OVER";delay(6000);

getch();

exit(0);

}

}

}

}

else

continue;

}

}

}

}//CASE 3

break;

}//SWITCH

getch();

}

//2048

class game

{

int a[4][4];

char opt;

int st,q;

public:

game()

{

st=0;

}

void option(); /\*function to accept from user\*/

void control(char);/\*perform operations\*/

void endgame();/\*to check whether the user has won the game\*/

void check();

void intializeA();/\*randomly initialize numbers in the array\*/

void display();/\*display after every operation\*/

void check2();

void clearoff();/\*initailize all numbers to 0\*/

};

void won()

{

cleardevice();

settextstyle(3,0,100);

outtext("GAME");

outtext("WON");

}

void lose()

{

cleardevice();

settextstyle(3,0,100);

outtext("GAME");

outtext("OVER");

}

void game::control(char opt)

{

int i,j;

switch(opt)

{

case 'a':

{

int k=0;

for( i=0;i<4;i++)

{

k=0;

for(j=0;j<4;j++)

{

if(a[i][j]!=0)

a[i][k++]=a[i][j];

}

for(;k<4;k++)

{

a[i][k]=0;

}

for( j=1;j<4;j++)

{

if(a[i][j]==a[i][j-1])

{

a[i][j-1]+=a[i][j];

a[i][j]=0;

}

}

k=0;

for(j=0;j<4;++j)

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

a[i][k++]=a[i][j];

}

for(;k<4;k++)

{

a[i][k]=0;

}

}

break;

}

case 'd':

{

int k;

for(i=0;i<4;i++)

{

k=3;

for(j=3;j>=0;j--)

{

if(a[i][j]!=0)

a[i][k--]=a[i][j];

}

for(;k>=0;k--)

{

a[i][k]=0;

}

for(j=2;j>=0;j--)

{

if(a[i][j]==a[i][j+1])

{

a[i][j+1]+=a[i][j];

a[i][j]=0;

}

}

k=3;

for(j=3;j>=0;--j)

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

a[i][k--]=a[i][j];

}

for(;k>=0;k--)

{

a[i][k]=0;

}

}

break;

}

case 'w':

{

int k;

for(j=0;j<4;j++)

{

k=0;

for(i=0;i<4;i++)

{

if(a[i][j]!=0)

a[k++][j]=a[i][j];

}

for(;k<4;k++)

{

a[k][j]=0;

}

for(i=1;i<4;i++)

{

if(a[i][j]==a[i-1][j])

{

a[i-1][j]+=a[i][j];

a[i][j]=0;

}

}

k=0;

for(i=0;i<4;++i)

{

if(a[i][j]!=0)

a[k++][j]=a[i][j];

}

for(;k<4;k++)

{

a[k][j]=0;

}

}

break;

}

case 's':

{

int k;

for(j=0;j<4;j++)

{

k=3;

for(i=3;i>=0;i--)

{

if(a[i][j]!=0)

a[k--][j]=a[i][j];

}

for(;k>=0;k--)

{

a[k][j]=0;

}

for(i=2;i>=0;i--)

{

if(a[i][j]==a[i+1][j])

{

a[i+1][j]+=a[i][j];

a[i][j]=0;

}

}

k=3;

for(i=3;i>=0;i--)

{

if(a[i][j]!=0)

a[k--][j]=a[i][j];

}

for(;k>=0;k--)

{

a[k][j]=0;

}

}

break;

}

default:

exit(1);

}

endgame();

}

void game::intializeA()

{

int i,j,flag=0;

while(flag!=1)

{

i=random(4)+0;

j=random(4)+0;

if(a[i][j]==0)

{

a[i][j]=2;

flag=1;

}

}

option();

}

void game::endgame()

{

int i,j;

st=0;

for(i=0;i<4;i++)

for(j=0;j<4;j++)

{

if(a[i][j]==0)

{st=2;

break;}

}

for(i=0;i<4;i++)

for(j=0;j<4;j++)

{

if(a[i][j]==2048)

{st=1;

break;}

}

check();

}

void game ::option()

{

opt=getch();

control(opt);

}

void game::check()

{

if(st==1)

{

won();

delay(1000);

exit(0);

}

if(st!=2&&st!=1)

{lose();

delay(1000);

exit(0);

}

else

check2();

}

void game::check2()

{

display();

}

void game::display()

{

cleardevice();

for(int i=0;i<4;i++)

{

for(int j=0;j<4;j++)

{

if(a[i][j]==0)

bmp\_load(150+85\*j,100+85\*i,"main\\bl.bmp");

else if(a[i][j]==2)

bmp\_load(150+85\*j,100+85\*i,"main\\2.bmp");

else if(a[i][j]==4)

bmp\_load(150+85\*j,100+85\*i,"main\\4.bmp");

else if(a[i][j]==8)

bmp\_load(150+85\*j,100+85\*i,"main\\8.bmp");

else if(a[i][j]==16)

bmp\_load(150+85\*j,100+85\*i,"main\\16.bmp");

else if(a[i][j]==32)

bmp\_load(150+85\*j,100+85\*i,"main\\32.bmp");

else if(a[i][j]==64)

bmp\_load(150+85\*j,100+85\*i,"main\\64.bmp");

else if(a[i][j]==128)

bmp\_load(150+85\*j,100+85\*i,"main\\128.bmp");

else if(a[i][j]==256)

bmp\_load(150+85\*j,100+85\*i,"main\\256.bmp");

else if(a[i][j]==512)

bmp\_load(150+85\*j,100+85\*i,"main\\512.bmp");

else if(a[i][j]==1024)

bmp\_load(150+85\*j,100+85\*i,"main\\1024.bmp");

else if(a[i][j]==2048)

bmp\_load(150+85\*j,100+85\*i,"main\\2048.bmp");

}

}

intializeA();

}

void game::clearoff()

{

int i,j;

for(i=0;i<4;i++)

for(j=0;j<4;j++)

{

a[i][j]=0;

}

}

void t048()

{

game g;

clrscr();

cleardevice();

cout<<"Welcome to 2048 game!\nRules of the game:\nThe objective of the game is to get the number 2048 using the additions of the number 2 and its multiples.\nYou will have a grid of 16 tiles. A number 2 will be given initially. Move up, down, left, or right trying to join 2 equal numbers.\nWhen 2 equal numbers are in touch they add up. Each turn a new 2 is provided somewhere in the grid. If there are no free tiles on our grid, the game ends.\nBy adding numbers, we get higher numbers and we can approach to 2048, which is the goal of the game.\nControls:\nThe 'W', 'A', 'S' and 'D' keys act as arrow keys and any other key pressed will exit the game. Press 'S' key to continue...";

g.clearoff();

g.intializeA();

g.display();

g.option();

getch();

}

//BLOXORZ

int ax1=100,ay1=100,ax2=138,ay2=138;

int p=ax1+38,q=ay1+38,r=ax2+38,s=ay2+38;

int result(int a,int b,int c,int d)

{

if(a<100||a>480||b<100||b>440||c<100||c>480||d<100||d>440)

{

lose();

return(1);

}

else if(a==p&&b==q&&c==r&&d==s)

{

won();

return(2);

}

else

return(0);

}

void original()

{

setcolor(WHITE);

line(100,100,480,100);

line(100,100,100,404);

line(100,404,480,404);

line(480,404,480,100);

line(138,100,138,404);

line(176,100,176,404);

line(214,100,214,404);

line(252,100,252,404);

line(290,100,290,404);

line(328,100,328,404);

line(366,100,366,404);

line(404,100,404,404);

line(442,100,442,404);

line(100,100,480,100);

line(100,138,480,138);

line(100,176,480,176);

line(100,214,480,214);

line(100,252,480,252);

line(100,290,480,290);

line(100,328,480,328);

line(100,366,480,366);

setfillstyle(1,BLUE);

}

void bloxorz()

{

clrscr();

int gdriver = DETECT, gmode,t;

initgraph(&gdriver, &gmode,"..\\bgi");

char f,ch;

cout<<"Welcome to Bloxorz!\nRules:\nMain objective of the game is to put the block into the hole.\n The block can be moved using arrow keys 'W', 'A', 'X', and 'D' and 'S' key exits the game..\nPress any key to continue...";

getch();

cleardevice();

setcolor(WHITE);

line(100,100,480,100);

line(100,100,100,404);

line(100,404,480,404);

line(480,404,480,100);

line(138,100,138,404);

line(176,100,176,404);

line(214,100,214,404);

line(252,100,252,404);

line(290,100,290,404);

line(328,100,328,404);

line(366,100,366,404);

line(404,100,404,404);

line(442,100,442,404);

line(100,100,480,100);

line(100,138,480,138);

line(100,176,480,176);

line(100,214,480,214);

line(100,252,480,252);

line(100,290,480,290);

line(100,328,480,328);

line(100,366,480,366);

while(t!=1&&t!=2)

{

setfillstyle(1,RED);

bar(p,q,r,s);

setfillstyle(1,YELLOW);

bar(ax1,ay1,ax2,ay2);

f=getch();

if(f=='d')

{

if(ax2-ax1==38||ax2-ax1==-38)

{

if(ay2-ay1==76)

{

cleardevice();

original();

bar(ax1+=38,ay1,ax2+=38,ay2);

t=result(ax1,ay1,ax2,ay2);

}

else

{

cleardevice();

original();

bar(ax1+=38,ay1,ax2+=76,ay2);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(ax2-ax1==76||ax2-ax1==-76)

{

cleardevice();

original();

bar(ax1+=76,ay1,ax2+=38,ay2);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(f=='a')

{

if(ax2-ax1==38||ax2-ax1==-38)

{

if(ay2-ay1==76)

{

cleardevice();

original();

bar(ax1-=38,ay1,ax2-=38,ay2);

t=result(ax1,ay1,ax2,ay2);

}

else

{

cleardevice();

original();

bar(ax1-=76,ay1,ax2-=38,ay2);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(ax2-ax1==-76||ax2-ax1==76)

{

cleardevice();

original();

bar(ax1-=38,ay1,ax2-=76,ay2);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(f=='x')

{

if(ay2-ay1==38)

{

if(ax2-ax1==76)

{

cleardevice();

original();

bar(ax1,ay1+=38,ax2,ay2+=38);

t=result(ax1,ay1,ax2,ay2);

}

else

{

cleardevice();

original();

bar(ax1,ay1+=38,ax2,ay2+=76);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(ay2-ay1==76)

{

cleardevice();

original();

bar(ax1,ay1+=76,ax2,ay2+=38);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(f=='w')

{

if(ay2-ay1==38)

{

if(ax2-ax1==76)

{

cleardevice();

original();

bar(ax1,ay1-=38,ax2,ay2-=38);

t=result(ax1,ay1,ax2,ay2);

}

else

{

cleardevice();

original();

bar(ax1,ay1-=76,ax2,ay2-=38);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(ay2-ay1==76)

{

cleardevice();

original();

bar(ax1,ay1-=38,ax2,ay2-=76);

t=result(ax1,ay1,ax2,ay2);

}

}

else if(f=='s')

exit(1);

}

getch();

}

//SOLITAIRE

struct card

{

char deck[24][4];

char stack[4][13][4];

char table[20][7][5];

int top[4],t[7],dt;

}c;

char tmp[4]; int tp[2]; int n;

int kp[2]={0,0};

void initialise()

{

strcpy(c.deck[0],"7c");

strcpy(c.deck[1],"Qc");

strcpy(c.deck[2],"9s");

strcpy(c.deck[3],"5c");

strcpy(c.deck[4],"3h");

strcpy(c.deck[5],"8c");

strcpy(c.deck[6],"5s");

strcpy(c.deck[7],"Ks");

strcpy(c.deck[8],"9d");

strcpy(c.deck[9],"4s");

strcpy(c.deck[10],"3d");

strcpy(c.deck[11],"Jc");

strcpy(c.deck[12],"10h");

strcpy(c.deck[13],"3c");

strcpy(c.deck[14],"4h");

strcpy(c.deck[15],"Qs");

strcpy(c.deck[16],"4c");

strcpy(c.deck[17],"Jd");

strcpy(c.deck[18],"2h");

strcpy(c.deck[19],"2c");

strcpy(c.deck[20],"Kc");

strcpy(c.deck[21],"5d");

strcpy(c.deck[22],"Qh");

strcpy(c.deck[23],"Kd");

strcpy(c.table[0][0],"18s");

strcpy(c.table[0][1],"04d");

strcpy(c.table[1][1],"1Ah");

strcpy(c.table[0][2],"0Qd");

strcpy(c.table[1][2],"06h");

strcpy(c.table[2][2],"17s");

strcpy(c.table[0][3],"06d");

strcpy(c.table[1][3],"03s");

strcpy(c.table[2][3],"010c");

strcpy(c.table[3][3],"1Ad");

strcpy(c.table[0][4],"09c");

strcpy(c.table[1][4],"0Jh");

strcpy(c.table[2][4],"02s");

strcpy(c.table[3][4],"09h");

strcpy(c.table[4][4],"16c");

strcpy(c.table[0][5],"08h");

strcpy(c.table[1][5],"010s");

strcpy(c.table[2][5],"06s");

strcpy(c.table[3][5],"0Kh");

strcpy(c.table[4][5],"08d");

strcpy(c.table[5][5],"17h");

strcpy(c.table[0][6],"010d");

strcpy(c.table[1][6],"05h");

strcpy(c.table[2][6],"02d");

strcpy(c.table[3][6],"07d");

strcpy(c.table[4][6],"0Js");

strcpy(c.table[5][6],"0As");

strcpy(c.table[6][6],"1Ac");

c.top[0]=0;

c.top[1]=0;

c.top[2]=0;

c.top[3]=0;

c.t[0]=0;

c.t[1]=1;

c.t[2]=2;

c.t[3]=3;

c.t[4]=4;

c.t[5]=5;

c.t[6]=6;

c.dt=-1;

}

char\* filename(char temp[5])

{

char name[20];

strcpy(name,"main//");

strcat(name,temp);

strcat(name,".bmp");

return name;

}

int wincheck()

{

if(c.stack[0][c.top[0]-1][0]=='K'&&c.stack[1][c.top[1]-1][0]=='K'&&c.stack[2][c.top[2]-1][0]=='K'&&c.stack[3][c.top[3]-1][0]=='K')

return 1;

else return 0;

}

void display()

{

char temp[3];

if(c.dt!=23)

bmp\_load(10,10,"main//deck.bmp");

else

bmp\_load(10,10,"main//last.bmp");

for(int i=0;i<4;i++)

{

if(c.top[i]==0)

bmp\_load(279+(i\*90),10,"main//blank.bmp");

else

bmp\_load(279+(i\*90),10,filename(c.stack[i][c.top[i]-1]));

}

for(int j=0;j<7;j++)

{

if(c.t[j]==-1)

bmp\_load(90\*j+10,160,"main//blank.bmp");

for(i=0;i<=c.t[j];i++)

{

if(c.table[i][j][0]=='0')

bmp\_load(90\*j+10,160+20\*i,"main//deck.bmp");

else

{

temp[0]=c.table[i][j][1];

temp[1]=c.table[i][j][2];

temp[2]=c.table[i][j][3];

temp[3]=c.table[i][j][4];

bmp\_load(90\*j+10,160+20\*i,filename(temp));

}

}

}

if(c.dt>=0)

{

bmp\_load(100,10,filename(c.deck[c.dt]));

}

}

void drawcursor(int kp[2])

{

setcolor(8);

if(kp[0]==0)

{

line(kp[1]\*90+5,0,kp[1]\*90+5,120);

line(kp[1]\*90+5,120,kp[1]\*90+94,120);

line(kp[1]\*90+94,120,kp[1]\*90+94,0);

line(kp[1]\*90+94,0,kp[1]\*90+5,0);

}

else

{

if(c.t[kp[1]]==-1)

{

line(kp[1]\*90+5,150,kp[1]\*90+5,270);

line(kp[1]\*90+5,270,kp[1]\*90+95,270);

line(kp[1]\*90+95,270,kp[1]\*90+95,150);

line(kp[1]\*90+95,150,kp[1]\*90+5,150);

}

else

{

line(kp[1]\*90+5,150,kp[1]\*90+5,270+(c.t[kp[1]])\*20);

line(kp[1]\*90+5,270+(c.t[kp[1]])\*20,kp[1]\*90+95,270+(c.t[kp[1]])\*20);

line(kp[1]\*90+95,270+(c.t[kp[1]])\*20,kp[1]\*90+95,150);

line(kp[1]\*90+95,150,kp[1]\*90+5,150);

}

}

if(tp[0]!=-1&&tp[1]!=-1)

{

setcolor(4);

if(tp[0]==1)

{

line(tp[1]\*90+5,150,tp[1]\*90+5,270+(c.t[tp[1]])\*20);

line(tp[1]\*90+5,270+(c.t[tp[1]])\*20,tp[1]\*90+95,270+(c.t[tp[1]])\*20);

line(tp[1]\*90+95,270+(c.t[tp[1]])\*20,tp[1]\*90+95,150);

line(tp[1]\*90+95,150,tp[1]\*90+5,150);

}

else

{

line(tp[1]\*90+5,0,tp[1]\*90+5,120);

line(tp[1]\*90+5,120,tp[1]\*90+94,120);

line(tp[1]\*90+94,120,tp[1]\*90+94,0);

line(tp[1]\*90+94,0,tp[1]\*90+5,0);

}

}

}

int color(char a[5],char b[5])

{

int m,n,i,j;

if(tp[0]==0&&tp[1]==1)

{

if(a[1]=='s'||a[1]=='h'||a[1]=='c'||a[1]=='d') i=1;

else if(a[2]=='s'||a[2]=='h'||a[2]=='c'||a[2]=='d') i=2;

else if(a[3]=='s'||a[3]=='h'||a[3]=='c'||a[3]=='d') i=3;

if(b[1]=='s'||b[1]=='h'||b[1]=='c'||b[1]=='d') j=1;

else if(b[2]=='s'||b[2]=='h'||b[2]=='c'||b[2]=='d') j=2;

else if(b[3]=='s'||b[3]=='h'||b[3]=='c'||b[3]=='d') j=3;

if(a[i]=='h'||a[i]=='d') m=0;

else m=1;

if(b[j]=='h'||b[j]=='d') n=0;

else n=1;

if(m==n) return (1);

else return (0);

}

else

{

if(a[2]=='0') i=3;

else i=2;

if(b[2]=='0') j=3;

else j=2;

if(a[i]=='h'||a[i]=='d') m=0;

else if(a[i]=='s'||a[i]=='c') m=1;

if(b[i]=='h'||b[j]=='d') n=0;

else if(b[i]=='s'||b[j]=='c') n=1;

if(m==n) return (1);

else return (0);

}

}

void deckopt(int kp[2])

{

if(kp[0]==0&&kp[1]==0)

{

c.dt++;

if(c.dt>=n)

c.dt=-1;

}

}

void solitaire()

{

init\_graph();

char ch;

tmp[0]='0'; tp[0]=-1; tp[1]=-1; int posn; n=24;

cout<<"Welcome to Solitaire Game!\nRules:\nThere is a total of 52 cards in the game.\nThe first objective is to release and play into position certain cards to build up each foundation, in sequence and in suit, from the ace through the king. The ultimate objective is to build the whole pack onto the foundations, and if that can be done, the Solitaire game is won.\n The rank of Cards from high to low is K>Q>J>10>9>8>7>6>5>4>3>2>A\nThe 4 different types of piles are:\n1. The Tableau: Seven piles that make up the main table.\n2. The Foundations: Four piles on which a whole suit or sequence must be built up. The 4 aces are the bottom card or base of the foundations. The foundation piles are hearts, diamonds, spades and clubs.\n3. The Stock or Hand pile: If the entire is not laid out in a tableau at the begining of a game, the remaining cards form the stock pile from which additional cards are brought into play according to the rules.\n4. The Talon or Waste pile:";

cout<<" Cards from the stock pile that have no place in tableau or foundations are laid face up in the waste pile.\nTransfer the cards from the waste pile to the tableau and from the tableau to the foundations. Cards can be arranged in the tableau in decreasing order of the rank and if they have alternate colors. These cards are further moved to the foundations by piling from A-K.\nControls:\nThe 'W', 'A', 'S' and 'D' keys act as the arrow keys used to move the curser.\n'X' key is used to exit\n'N' key is used to select/drop the card from one pile to another\nPress any key to continue....";

getch();

cleardevice();

initialise();

l1:

clrscr();

display();

drawcursor(kp);

ch=getch();

if(ch=='d')

{

kp[1]++;

if(kp[0]==0&&kp[1]==2)

kp[1]++;

if(kp[1]==7)

kp[1]=0;

}

else if(ch=='a')

{

kp[1]--;

if(kp[0]==0&&kp[1]==2)

kp[1]--;

if(kp[1]==-1)

kp[1]=6;

}

else if(ch=='s'&&kp[0]!=1)

{

kp[0]++;

}

else if(ch=='w'&&kp[0]!=0)

{

kp[0]--;

if(kp[0]==0&&kp[1]==2)

kp[0]++;

}

else if(ch=='n'&&kp[0]==1&&tmp[0]=='0'&&c.t[kp[1]]!=-1)

{

tmp[0]=c.table[c.t[kp[1]]][kp[1]][1];

tmp[1]=c.table[c.t[kp[1]]][kp[1]][2];

tmp[2]=c.table[c.t[kp[1]]][kp[1]][3];

tmp[3]=c.table[c.t[kp[1]]][kp[1]][4];

tp[0]=kp[0];

tp[1]=kp[1];

}

else if(ch=='n'&&kp[0]==0&&kp[1]==1&&tmp[0]=='0'&&c.dt!=-1)

{

tmp[0]=c.deck[c.dt][0];

tmp[1]=c.deck[c.dt][1];

tmp[2]=c.deck[c.dt][2];

tmp[3]=c.deck[c.dt][3];

tp[0]=kp[0];

tp[1]=kp[1];

}

else if(ch=='n'&&kp[0]==0&&kp[1]==0&&tmp[0]=='0')

deckopt(kp);

else if(ch=='n'&&kp[0]==1&&tp[0]==0&&tp[1]==1&&tmp[0]!='0'&&tmp[0]=='K'&&c.t[kp[1]]==-1)

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

c.table[++c.t[kp[1]]][kp[1]][0]='1';

c.table[c.t[kp[1]]][kp[1]][1]=tmp[0];

c.table[c.t[kp[1]]][kp[1]][2]=tmp[1];

c.table[c.t[kp[1]]][kp[1]][3]=tmp[2];

c.table[c.t[kp[1]]][kp[1]][4]=tmp[3];

tmp[0]='0';

tp[0]=-1; tp[1]=-1;

}

else if(ch=='n'&&kp[0]==1&&tp[0]==0&&tp[1]==1&&tmp[0]!='0'&&color(c.table[c.t[kp[1]]][kp[1]],tmp)==0)

{

if((c.table[c.t[kp[1]]][kp[1]][1]=='2'&&tmp[0]=='A')||(c.table[c.t[kp[1]]][kp[1]][1]=='3'&&tmp[0]=='2')||(c.table[c.t[kp[1]]][kp[1]][1]=='4'&&tmp[0]=='3')||(c.table[c.t[kp[1]]][kp[1]][1]=='5'&&tmp[0]=='4')||(c.table[c.t[kp[1]]][kp[1]][1]=='6'&&tmp[0]=='5')||(c.table[c.t[kp[1]]][kp[1]][1]=='7'&&tmp[0]=='6')||(c.table[c.t[kp[1]]][kp[1]][1]=='8'&&tmp[0]=='7')||(c.table[c.t[kp[1]]][kp[1]][1]=='9'&&tmp[0]=='8')||(c.table[c.t[kp[1]]][kp[1]][1]=='1'&&tmp[0]=='9')||(c.table[c.t[kp[1]]][kp[1]][1]=='J'&&tmp[0]=='1')||(c.table[c.t[kp[1]]][kp[1]][1]=='Q'&&tmp[0]=='J')||(c.table[c.t[kp[1]]][kp[1]][1]=='K'&&tmp[0]=='Q') )

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

c.table[++c.t[kp[1]]][kp[1]][0]='1';

c.table[c.t[kp[1]]][kp[1]][1]=tmp[0];

c.table[c.t[kp[1]]][kp[1]][2]=tmp[1];

c.table[c.t[kp[1]]][kp[1]][3]=tmp[2];

c.table[c.t[kp[1]]][kp[1]][4]=tmp[3];

}

tmp[0]='0';

tp[0]=-1; tp[1]=-1;

}

else if(ch=='n'&&kp[0]==1&&tp[0]==1&&tmp[0]!='0')

{

if(c.table[c.t[tp[1]]][tp[1]][0]=='1'&&c.table[c.t[tp[1]]][tp[1]][1]=='K'&&c.t[kp[1]]==-1)

{

c.t[kp[1]]++;

strcpy(c.table[c.t[kp[1]]][kp[1]],c.table[c.t[tp[1]]][tp[1]]);

c.t[tp[1]]--;

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else

{

posn=-1;

for(int i=0;i<=c.t[tp[1]];i++)

if(c.table[i][tp[1]][0]=='1'&&color(c.table[i][tp[1]],c.table[c.t[kp[1]]][kp[1]])==0)

{

if((c.table[i][tp[1]][1]=='A'&&c.table[c.t[kp[1]]][kp[1]][1]=='2')||(c.table[i][tp[1]][1]=='2'&&c.table[c.t[kp[1]]][kp[1]][1]=='3')||(c.table[i][tp[1]][1]=='3'&&c.table[c.t[kp[1]]][kp[1]][1]=='4')||(c.table[i][tp[1]][1]=='4'&&c.table[c.t[kp[1]]][kp[1]][1]=='5')||(c.table[i][tp[1]][1]=='5'&&c.table[c.t[kp[1]]][kp[1]][1]=='6')||(c.table[i][tp[1]][1]=='6'&&c.table[c.t[kp[1]]][kp[1]][1]=='7')||(c.table[i][tp[1]][1]=='7'&&c.table[c.t[kp[1]]][kp[1]][1]=='8')||(c.table[i][tp[1]][1]=='8'&&c.table[c.t[kp[1]]][kp[1]][1]=='9')||(c.table[i][tp[1]][1]=='9'&&c.table[c.t[kp[1]]][kp[1]][1]=='1')||(c.table[i][tp[1]][1]=='1'&&c.table[c.t[kp[1]]][kp[1]][1]=='J')||(c.table[i][tp[1]][1]=='J'&&c.table[c.t[kp[1]]][kp[1]][1]=='Q')||(c.table[i][tp[1]][1]=='Q'&&c.table[c.t[kp[1]]][kp[1]][1]=='K'))

{

posn=i;

break;

}

}

if(posn!=-1)

{

for(int j=posn;j<=c.t[tp[1]];j++)

strcpy(c.table[++c.t[kp[1]]][kp[1]],c.table[j][tp[1]]);

c.t[tp[1]]-=c.t[tp[1]]-posn+1;

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

}

tmp[0]='0';

tp[0]=-1; tp[1]=-1;

}

else if(ch=='n'&&kp[0]==0&&kp[1]>1&&tmp[0]!='0')

{

if(tp[0]==1)

{

if(c.top[kp[1]-3]==0&&tmp[0]=='A')

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==1&&tmp[0]=='2'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==2&&tmp[0]=='3'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==3&&tmp[0]=='4'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==4&&tmp[0]=='5'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==5&&tmp[0]=='6'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==6&&tmp[0]=='7'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==7&&tmp[0]=='8'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==8&&tmp[0]=='9'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==9&&tmp[0]=='1'&&tmp[1]=='0'&&tmp[2]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==10&&tmp[0]=='J'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][2])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==11&&tmp[0]=='Q'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

else if(c.top[kp[1]-3]==12&&tmp[0]=='K'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

c.t[tp[1]]--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

c.table[c.t[tp[1]]][tp[1]][0]='1';

}

}

else if(n!=0)

{

if(c.top[kp[1]-3]==0&&tmp[0]=='A')

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==1&&tmp[0]=='2'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==2&&tmp[0]=='3'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==3&&tmp[0]=='4'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==4&&tmp[0]=='5'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==5&&tmp[0]=='6'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==6&&tmp[0]=='7'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==7&&tmp[0]=='8'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==8&&tmp[0]=='9'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==9&&tmp[0]=='1'&&tmp[1]=='0'&&tmp[2]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==10&&tmp[0]=='J'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][2])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==11&&tmp[0]=='Q'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

else if(c.top[kp[1]-3]==12&&tmp[0]=='K'&&tmp[1]==c.stack[kp[1]-3][c.top[kp[1]-3]-1][1])

{

for(int i=c.dt;i<n;i++)

strcpy(c.deck[i],c.deck[i+1]);

n--;

c.dt--;

strcpy(c.stack[kp[1]-3][c.top[kp[1]-3]++],tmp);

}

}

tmp[0]='0';

tp[0]=-1; tp[1]=-1;

}

else if(ch=='x')

exit(1);

if(wincheck()==1)

{

won();

ch=getch();

exit(1);

}

else

goto l1;

cin.get();

closegraph();

}

void main()

{

int c;

init\_graph();

cleardevice();

setcolor(RED);

settextstyle(10, HORIZ\_DIR,3);

cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n ";

outtextxy(10,0,"Menu");

outtextxy(10,40,"1. Solitaire");

outtextxy(10,80,"2. Bloxorz");

outtextxy(10,120,"3. 2048");

outtextxy(10,160,"4. One more line");

outtextxy(10,200,"Enter your choice: ");

cin>>c;

if(c==1)

solitaire();

else if(c==2)

bloxorz();

else if(c==3)

t048();

else if(c==4)

onemoreline();

else

exit(1);

getch();

}

int init\_graph()

{

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode,"..\\bgi");

errorcode = graphresult();

if (errorcode != grOk) /\* an error occurred \*/

{

cout << "Error in graphical mode:\n" << grapherrormsg(errorcode) << endl;

system ("pause");

exit(1); /\* return with error code \*/

}

return grOk;

}

//Bitmap loading

RGBquad \*bmp\_load(int x1, int y1, char \*filename)

{

register int i,j;

int x2, y2;

unsigned char c;

bmpfileheader bmph;

bmpinfoheader bmpi;

RGBquad \*rgb;

ifstream img(filename,ios::in|ios::binary);

if (!img) {

cout << "The file " << filename << " does not exist." << endl;

exit(EXIT\_FAILURE);

}

img.read((char \*)&bmph,sizeof(bmph));

img.read((char \*)&bmpi,sizeof(bmpi));

if((bmph.type[0]!='B') || (bmph.type[1]!='M')) {

cout << "BMP file does not have a header BM." << endl;

img.close();

exit(EXIT\_FAILURE);

};

if((bmph.reserved[0] != '\0') ||

(bmph.reserved[1] != '\0') ||

(bmph.reserved[2] != '\0') ||

(bmph.reserved[3] != '\0'))

{

cout << "BMP file does not have the field bmph.reserved free." << endl;

img.close();

exit(EXIT\_FAILURE);

};

if (bmpi.planes != 1)

{

cout << "BMP file has bmpi.planes != 1" << endl;

img.close();

exit(EXIT\_FAILURE);

}

if (bmpi.compression!=0) {

cout << "Compressed BMP File" << endl;

cout << "Compressed BMP Files can't be read " << endl;

img.close();

exit(EXIT\_FAILURE);

}

switch (bmpi.bitcount)

{

case 1: {

cout << "Color format monochrome still not supported" << endl;

img.close();

exit(EXIT\_FAILURE);

break;

}

case 4:

if (getmaxcolor() < 15)

{

cout << "The current configuration does not allow 16 colors." << endl;

exit(EXIT\_FAILURE);

}

rgb = new RGBquad[1<<bmpi.bitcount];

img.read((char \*)rgb,sizeof(\*rgb)\*(1<<bmpi.bitcount));

x2 = x1 + (int)bmpi.width;

y2 = y1 + (int)bmpi.height;

for(j=y2-1; j>=y1; j--) { //Fix Photo j=y2

for(i=x1; i<x2; i++) { //Fix Photo i<=x2

img.read((char \*)&c,sizeof(c));

putpixel(i,j,c>>4);

i++;

putpixel(i,j,(c & 15)); //c<<4)>>4);

}

}

break;

case 8:

cout << "256 Color Format not supported." << endl;

img.close();

exit(EXIT\_FAILURE);

break;

case 24:

cout << "24 Color Format not supported." << endl;

img.close();

exit(EXIT\_FAILURE);

break;

default:

cout << "BMP file invalid" << endl;

img.close();

exit(EXIT\_FAILURE);

};

// img.close();

return rgb;

};