#include<iostream>

#include<conio.h>

#include<string>

#include<windows.h>

#include<ctime>

using namespace std;

void main()

{

srand(time(NULL));

int n, i, c = 0, r = 0, s = 0, col, row, col1, row1, check = 1, count = 1, check1 = 0, check2 = 0, check3 = 0, n1, n2 = 1, n3 = 0, shape = 0, index;

string click = "c", clicked = "c";

int dec[52], b1[20][1], b2[8];

string b[28], de[52], deck[24], board[20][8], f[5][14];

string a[52] = { "A\3","2\3","3\3","4\3","5\3","6\3","7\3","8\3","9\3","10\3","J\3","Q\3","K\3",

"A\4","2\4","3\4","4\4","5\4","6\4","7\4","8\4","9\4","10\4","J\4","Q\4","K\4",

"A\5","2\5","3\5","4\5","5\5","6\5","7\5","8\5","9\5","10\5","J\5","Q\5","K\5",

"A\6","2\6","3\6","4\6","5\6","6\6","7\6","8\6","9\6","10\6","J\6","Q\6","K\6" };

string x, card, card1, status = "lose", series[14], cards[14];

string hearts[13], diamonds[13], clubs[13], spades[13];

int o = 0, num = 0, num1 = 0, s1 = 1, moves = 0, check4 = 0, hide[8];

cout << "\t\t \*\*\*\*\*\*\*\*\* SOLITAIRE \*\*\*\*\*\*\*\*\n";

cout << "\n\n\n\n\n\n\t\t\tloading ";

for (int i = 0; i < 5; i++)

{

cout << ".";

Sleep(300);

}

system("CLS");

for (i = 0; i < 52; i++)

{

if (o < 13)

{

if (i < 13)

hearts[o] = a[i];

else if (i > 12 && i < 26)

diamonds[o] = a[i];

else if (i > 25 && i < 39)

clubs[o] = a[i];

else if (i > 38 && i < 52)

spades[o] = a[i];

o++;

if (o == 13)

{

o = 0;

}

}

}

for (int i = 0; i<52; i++)

{

dec[i] = rand() % 52;

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

if (dec[j] == dec[i])

i--;

}

for (i = 0; i<52; i++)

{

n = dec[i];

de[i] = a[n];

}

for (i = 0; i<24; i++)

{

deck[i] = de[i];

}

for (i = 24; i<52; i++)

{

if (r<28)

{

b[r] = de[i];

}

r++;

}

for (i = 0; i < 20; i++)

b1[i][0] = i;

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

b2[j] = j;

for (i = 1; i <= 13; i++)

{

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

{

if (s<28)

{

board[i][j] = b[s];

}

s++;

}

}

for (i = 1; i < 8; i++) //this array is for hidden cards

hide[i] = i - 1;

while (click != "q")

{

cout << "\t\t \*\*\*\*\*\*\*\*\* SOLITAIRE \*\*\*\*\*\*\*\*\n";

cout << "\t\t\t\t\t\t\t\t MOVES = " << moves;

cout << endl << " DECK \n";

for (i = 0; i < 24; i++)

cout << "(" << i << ") " << deck[i] << "\t";

cout << endl << endl << " FOUNDATION ";

f[1][0] = "\3";

f[2][0] = "\4";

f[3][0] = "\5";

f[4][0] = "\6";

cout << endl;

for (i = 1; i < 5; i++)

{

for (int j = 0; j < 14; j++)

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

cout << endl;

}

cout << " BOARD \n";

for (int j = 0; j < 8; j++)

if (j == 0)

cout << b2[j] << "\t";

else

cout << "|" << b2[j] << "|" << "\t";

cout << endl;

for (i = 1; i < 20; i++)

{

cout << b1[i][0] << ")\t";

for (int j = 1; j <= 7; j++)

{

check4 = 0;

for (int z = 0; z < 52; z++)

{

if (board[i][j] == a[z])

check4 = 1;

}

if (check4 == 1)

if (i>hide[j])

cout << board[i][j] << "\t";

else

cout << "X\t";

else

cout << " \t";

}

cout << endl;

}

cout << endl;

num = 0; num1 = 0;

cout << "From where you want to pick \nPRESS d for Deck \nPRESS b for Board \nPRESS f for foundation \nPRESS q to QUIT :\n";

cin >> click;

if (click == "d")

{

do

{

cout << "ENTER card no. :";

cin >> col;

} while (col > 23 || col<0);

card = deck[col];

}

else if (click == "b")

{

do {

cout << "ENTER column no. :";

cin >> col;

} while (col > 7 || col<1);

do {

cout << "ENTER row no. :";

cin >> row;

} while (row >19 || row<1);

num = 0; num1 = 0; s1 = 1;

for (int j = 0; j < 14; j++)

{

cards[j] = ' ';

series[j] = ' ';

}

count = row;

for (i = row; i <= count; i++) //check for board to pick up cards

{

check = 0;

for (int j = 0; j < 52; j++)

{

if (board[i][col] == a[j])

{

check = 1;

}

}

if (check == 1)

count++;

}

num = count - row;

for (int j = row; j <= i - 2; j++)

{

if (s1 <= num)

cards[s1] = board[j][col];

s1++;

}

n3 = i - 2;

card = cards[1];

for (i = 2; i <= num; i++)

{

for (int j = 0; j < 13; j++) //checking cards if in series

{

if (card == hearts[j] || card == diamonds[j])

{

if (cards[i] == clubs[j - 1] || cards[i] == spades[j - 1])

{

series[i] = cards[i];

card = cards[i];

num1++;

}

}

else if (card == clubs[j] || card == spades[j])

{

if (cards[i] == hearts[j - 1] || cards[i] == diamonds[j - 1])

{

series[i] = cards[i];

card = cards[i];

num1++;

}

}

}

}

series[1] = cards[1];

card = cards[1];

if (series[1] == cards[1])

num1++;

}

else if (click == "f")

{

do {

cout << "ENTER row no. :";

cin >> row;

} while (row > 4 || row < 1);

count = 1;

for (i = 1; i <= count; i++) //check for foundation to pick the last occuring card

{

check = 0;

for (int j = 0; j < 52; j++)

{

if (f[row][i] == a[j])

{

check = 1;

}

}

if (check == 1)

count++;

}

col = i - 2;

if (col > 0)

card = f[row][col];

else

click = "";

}

else

system("CLS");

if (click == "b" || click == "d" || click == "f")

if (card != " ")

if (num == num1)

{

row1 = row;

clicked = click;

col1 = col;

check1 = 0;

for (i = 0; i < 13; i++)

{

if (card == hearts[i] || card == diamonds[i]) //check1 is 1 for red and 0 for black

check1 = 1;

}

if (num1 <= 1)

{

cout << "You picked " << card << endl;

cout << "Where you want to place it :\nPRESS b for board:\nPRESS f for foundation :\nPRESS q to QUIT :\n";

}

else

{

cout << endl << "You picked : ";

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

cout << series[i] << " ";

cout << "\nWhere you want to place it :\nPRESS b for board:\nPRESS q to Quit :\n";

}

cin >> click;

if (click == "b")

{

cout << "ENTER column no. :";

cin >> col;

n2 = 1;

count = 1;

for (i = 1; i <= count; i++) //check for board to not overwrite

{

check = 0;

for (int j = 0; j < 52; j++)

{

if (board[i][col] == a[j])

{

check = 1;

}

}

if (check == 1)

count++;

}

if (i == 2)

{

if (card == "K\3" || card == "K\4" || card == "K\5" || card == "K\6") //for K rule in board

{

if (num1 <= 1)

{

board[1][col] = card;

if (clicked == "b")

board[row1][col1] = ' ';

else if (clicked == "d")

deck[col1] = ' ';

else if (clicked == "f")

f[row1][col1] = ' ';

}

else

{

for (int j = 1; j <= num1; j++)

board[j][col] = series[j];

for (int j = row; j <= n3; j++)

board[j][col1] = ' ';

}

if (hide[col1] > row1 - 2)

hide[col1] = row1 - 2;

if (hide[col1] < 1)

hide[col1] = 0;

moves++;

}

}

else

{

check2 = 0;

for (int j = 0; j < 13; j++)

{

if (board[i - 2][col] == hearts[j] || board[i - 2][col] == diamonds[j]) //check2 is 1 for red and 0 for black

{

if (j > 0)

{

n1 = j;

check2 = 1;

}

else

system("CLS");

}

else if (board[i - 2][col] == clubs[j] || board[i - 2][col] == spades[j])

{

if (j > 0)

{

n1 = j;

check2 = 0;

}

else

system("CLS");

}

}

if (check == 0)

{

if (check1 == 0 && check2 == 1)

{

if (card == clubs[n1 - 1] || card == spades[n1 - 1]) //check for lesser card

{

if (num1 <= 1)

{

board[i - 1][col] = card;

moves++;

}

else

for (int j = i - 1; j <= num1 + (i - 1); j++) //placing the series of cards

{

if (n2 <= num1)

{

board[j][col] = series[n2];

moves++;

if (hide[col1] > row1 - 2)

hide[col1] = row1 - 2;

if (hide[col1] < 1)

hide[col1] = 0;

}

n2++;

}

if (clicked == "b")

if (num1 <= 1)

{

board[row1][col1] = ' ';

if (hide[col1] > row1 - 2)

hide[col1] = row1 - 2;

if (hide[col1] < 1)

hide[col1] = 0;

}

else

for (int j = row; j <= n3; j++)

board[j][col1] = ' ';

else if (clicked == "d")

deck[col1] = ' ';

else if (clicked == "f")

f[row1][col1] = ' ';

}

}

else if (check1 == 1 && check2 == 0)

{

if (card == hearts[n1 - 1] || card == diamonds[n1 - 1]) //check for lesser card

{

if (num1 <= 1)

{

board[i - 1][col] = card;

moves++;

}

else

for (int j = i - 1; j <= num1 + (i - 1); j++) //placing the series of cards

{

if (n2 <= num1)

{

board[j][col] = series[n2];

moves++;

if (hide[col1] > row1 - 2)

hide[col1] = row1 - 2;

if (hide[col1] < 1)

hide[col1] = 0;

}

n2++;

}

if (clicked == "b")

if (num1 <= 1)

{

board[row1][col1] = ' ';

if (hide[col1] > row1 - 2)

hide[col1] = row1 - 2;

if (hide[col1] < 1)

hide[col1] = 0;

}

else

for (int j = row; j <= n3; j++)

board[j][col1] = ' ';

else if (clicked == "d")

deck[col1] = ' ';

else if (clicked == "f")

f[row1][col1] = ' ';

}

}

else

system("CLS");

}

}

}

else if (click == "f" && num1 <= 1)

{

for (i = 0; i < 13; i++) //shape of card

{

if (card == hearts[i])

shape = 1;

else if (card == diamonds[i])

shape = 2;

else if (card == clubs[i])

shape = 3;

else if (card == spades[i])

shape = 4;

}

for (i = 1; i < 13; i++) //storing lower card as card1

{

index = i - 1;

if (card == hearts[i])

card1 = hearts[index];

else if (card == diamonds[i])

card1 = diamonds[index];

else if (card == clubs[i])

card1 = clubs[index];

else if (card == spades[i])

card1 = spades[index];

}

cout << "ENTER row no. :";

cin >> row;

if (shape == row)

{

count = 1;

for (i = 1; i <= count; i++) //check for foundation to not overwrite

{

check = 0;

for (int j = 0; j < 52; j++)

{

if (f[row][i] == a[j])

{

check = 1;

}

}

if (check == 1)

count++;

}

if (i == 2)

{

if (card == "A\3" || card == "A\4" || card == "A\5" || card == "A\6") //for A rule in foundation

{

f[shape][1] = card;

if (clicked == "b")

{

board[row1][col1] = ' ';

if (hide[col1] > row1 - 2)

hide[col1] = row1 - 2;

if (hide[col1] < 1)

hide[col1] = 0;

}

else if (clicked == "d")

deck[col1] = ' ';

moves++;

}

}

if (i >= 3)

{

if (f[shape][i - 2] == card1) //checking the previous card

{

f[shape][i - 1] = card;

moves++;

if (clicked == "b")

{

board[row1][col1] = ' ';

if (hide[col1] > row1 - 2)

hide[col1] = row1 - 2;

if (hide[col1] < 1)

hide[col1] = 0;

}

else if (clicked == "d")

deck[col1] = ' ';

}

}

}

else

system("CLS");

}

else

system("CLS");

}

check3 = 0; //check for status

for (i = 1; i < 5; i++)

for (int j = 1; j < 14; j++)

for (n = 0; n < 52; n++)

if (f[i][j] == a[n])

check3++;

if (check3 == 52)

{

status = "win";

click = "q";

Sleep(500);

}

system("CLS");

}

do

{

if (status == "win")

cout << "\n\n\n\n\n\n\t\t\tYOU WON THE GAME \n\t\t\t";

if (status == "lose")

cout << "\n\n\n\n\n\n\t\t\t\tGAME OVER \n\t\t\t";

cout << "PRESS p to PLAY AGAIN\n\t\t\tPRESS e to EXIT :";

cin >> click;

system("CLS");

} while (click != "p" && click != "e");

if (click == "p")

main();

else

exit(100);

system("pause>null");

}