#include <stdio.h>

#include <time.h>

#include <stdlib.h>

//Define Variables for Arrays

#define card 13

#define suit 13

#define players 4

//Defining these here so i use these variables for alot of arrays, so i dont have to keep defining them

//Define Functions at start

void SaveGame();

void PlayGame();

void Display();

void NewGame();

void ContinueGame();

void Menu();

//----------------------------------------------------------------------------

//Main

void main()

{

//Declare Variables

int initialise;

//Title

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Welcome to the CardGame of War \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//Ask if starting a new game or loading one

printf("\nPlease Enter 1 to Load a Saved Game or any Other Number to Start a New Game:\t");

scanf("%d", &initialise);

//If Statement to use new or load game methods

if (initialise == 1)

{

//Go to Load Game Function

ContinueGame();

}

else

{

//Start a New Game

NewGame();

}

}

//Save Game- Here will write to file

void SaveGame(int playerCard[players][card], int playerSuit[players][suit], int numPlayers, int playerScores[players], int roundNum, int playerUsedCards[players][card])

{

//Declare File Names- Made a seperate one for scores to help stop mistakes/errors

FILE\* gameStatus;

FILE\* gameScores;

//Open & Write to files

gameStatus = fopen("GameSaved.txt", "w");

//Check if files are open

if (gameStatus == NULL)

{

printf("Sorry Game Status File could not be opened\n");

}

//If Both Files are open

else

{

//Print Number of Players and Round Number

fprintf(gameStatus, "%d %d\n\n", numPlayers, roundNum);

//Loop for each players Cards and Suits

for (int p = 0; p < numPlayers; p++)

{

//Only loop for 13 times- 13 cards each

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

{

//Print Card and Suit

fprintf(gameStatus, "%d %d\n", playerCard[p][i], playerSuit[p][i]);

}

//Skip a line before next player

fprintf(gameStatus, "\n");

}

//Loop for each players used cards

for (int q = 0; q < numPlayers; q++)

{

//Only loop for 13 times- 13 cards each

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

{

//Print Card if used or not

fprintf(gameStatus, "%d\n", playerUsedCards[q][j]);

}

//Skip a line before next player

fprintf(gameStatus, "\n");

}

//Close File and open other

fclose(gameStatus);

gameScores = fopen("GameScore.txt", "w");

//Check if File open

if (gameScores == NULL)

{

printf("Sorry Score File could not be opened\n");

}

else

{

//Only Loop for each Player

for (int p = 0; p < numPlayers; p++)

{

//Print each players scores to other file

fprintf(gameScores, "%d\n", playerScores[p]);

}

//Close File

fclose(gameScores);

}

}

}

//----------------------------------------------------------------------------

//Play Game- This is Seperate due to Continue Game Function

void PlayGame(int playerCard[players][card], int playerSuit[players][suit], int numPlayers, int playerScores[players], int roundNumber, int playerUsedCards[players][card])

{

//Assign variables

int cardChoice, player = 1, points = 0, chosenCard;

int currentPlayer, winner = 0, winningCard = 0;

int playerSelectedCard[4][4], playerSelectedSuit[4][4];

char exit = 'Y', round = 'Y';

//Game goes on for 13 rounds

do {

//Next person has atleast 1 turn option

do {

printf("\nRound %d", roundNumber);

//Update Current Player

currentPlayer = (player - 1);

//Display Player Cards

#pragma region DisplayCards

//Display Card Numbers

printf("\nPlayer %d's Cards are displayed below and number above them\n", player);

//Display Number of card

for (int sel1 = 0; sel1 < 13; sel1++)

{

if (sel1 < 9)

{

printf(" %d. ", (sel1 + 1));

}

else

{

printf(" %d. ", (sel1 + 1));

}

}

printf("\n");

//Display Top of Cards

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

{

printf(" \_\_\_\_\_\_ ");

}

printf("\n");

//Display Card Number at Top

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

{

//Card Played

if (playerCard[currentPlayer][c] == 0)

{

printf("| | ");

playerUsedCards[currentPlayer][c] = 1;

}

else if (playerCard[currentPlayer][c] > 1 && playerCard[currentPlayer][c] < 10)

{

printf("|%d | ", playerCard[currentPlayer][c]);

}

else if (playerCard[currentPlayer][c] == 1 || playerCard[currentPlayer][c] == 14)

{

printf("|A | ");

playerCard[currentPlayer][c] = 14;

}

else if (playerCard[currentPlayer][c] == 10)

{

printf("|%d | ", playerCard[currentPlayer][c]);

}

else if (playerCard[currentPlayer][c] == 11)

{

printf("|J | ");

}

else if (playerCard[currentPlayer][c] == 12)

{

printf("|Q | ");

}

else if (playerCard[currentPlayer][c] == 13)

{

printf("|K | ");

}

}

//Skip to new line

printf("\n");

//Display Card Suit

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

{

switch (playerSuit[currentPlayer][c])

{

case 0:

printf("| | ");

break;

case 1:

printf("| H | ");

break;

case 2:

printf("| D | ");

break;

case 3:

printf("| S | ");

break;

case 4:

printf("| C | ");

break;

}

}

//Skip to new line

printf("\n");

//Display Card Number at Bottom

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

{

if (playerCard[currentPlayer][c] == 0)

{

printf("|Played| ");

}

if (playerCard[currentPlayer][c] > 1 && playerCard[currentPlayer][c] < 10)

{

printf("| %d| ", playerCard[currentPlayer][c]);

}

else if (playerCard[currentPlayer][c] == 1 || playerCard[currentPlayer][c] == 14)

{

printf("| A| ");

}

else if (playerCard[currentPlayer][c] == 10)

{

printf("| %d| ", playerCard[currentPlayer][c]);

}

else if (playerCard[currentPlayer][c] == 11)

{

printf("| J| ");

}

else if (playerCard[currentPlayer][c] == 12)

{

printf("| Q| ");

}

else if (playerCard[currentPlayer][c] == 13)

{

printf("| K| ");

}

}

//Skip to new line

printf("\n");

//Display Bottom of Cards

for (int sel1 = 0; sel1 < 13; sel1++)

{

printf("|\_\_\_\_\_\_| ");

}

//Skip to new line

printf("\n");

#pragma endregion

//Ask Player to select card

printf("\nPlease Select Card you wish to Play (Or 0 to exit)\nChoice: ");

scanf("%d", &cardChoice);

//Set choice to players card

#pragma region SavePlayerChoice

//Check if trying to play a used card

while(playerUsedCards[currentPlayer][(cardChoice - 1)] == 1)

{

//Card already chosen

printf("\nCard Already Played!!!!!!");

//Ask Player to select card

printf("\nPlease Select Card you wish to Play (Or 0 to exit)\nChoice: ");

scanf("\t%d", &cardChoice);

}

//Save Player Select Card # and Suit

switch (cardChoice)

{

case 1:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][0];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][0];

//Set card to Played

playerCard[currentPlayer][0] = 0;

playerSuit[currentPlayer][0] = 0;

break;

case 2:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][1];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][1];

//Set card to Played

playerCard[currentPlayer][1] = 0;

playerSuit[currentPlayer][1] = 0;

break;

case 3:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][2];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][2];

//Set card to Played

playerCard[currentPlayer][2] = 0;

playerSuit[currentPlayer][2] = 0;

break;

case 4:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][3];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][3];

//Set card to Played

playerCard[currentPlayer][3] = 0;

playerSuit[currentPlayer][3] = 0;

break;

case 5:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][4];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][4];

//Set card to Played

playerCard[currentPlayer][4] = 0;

playerSuit[currentPlayer][4] = 0;

break;;

case 6:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][5];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][5];

//Set card to Played

playerCard[currentPlayer][5] = 0;

playerSuit[currentPlayer][5] = 0;

break;

case 7:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][6];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][6];

//Set card to Played

playerCard[currentPlayer][6] = 0;

playerSuit[currentPlayer][6] = 0;

break;

case 8:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][7];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][7];

//Set card to Played

playerCard[currentPlayer][7] = 0;

playerSuit[currentPlayer][7] = 0;

break;

case 9:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][8];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][8];

//Set card to Played

playerCard[currentPlayer][8] = 0;

playerSuit[currentPlayer][8] = 0;

break;

case 10:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][9];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][9];

//Set card to Played

playerCard[currentPlayer][9] = 0;

playerSuit[currentPlayer][9] = 0;

break;

case 11:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][10];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][10];

//Set card to Played

playerCard[currentPlayer][10] = 0;

playerSuit[currentPlayer][10] = 0;

break;

case 12:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][11];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][11];

//Set card to Played

playerCard[currentPlayer][11] = 0;

playerSuit[currentPlayer][11] = 0;

break;

case 13:

//Save Selected Card Details

playerSelectedCard[currentPlayer][currentPlayer] = playerCard[currentPlayer][12];

playerSelectedSuit[currentPlayer][currentPlayer] = playerSuit[currentPlayer][12];

//Set card to Played

playerCard[currentPlayer][12] = 0;

playerSuit[currentPlayer][12] = 0;

break;

}

#pragma endregion

//Add chosen card to points for round

#pragma region AddCardToPoints

//Save Card choice to Variable

chosenCard = playerSelectedCard[currentPlayer][currentPlayer];

//Set point decide by Card Choice

switch (chosenCard)

{

case 14:

points += 14;

break;

case 2:

points += 2;

break;

case 3:

points += 3;

break;

case 4:

points += 4;

break;

case 5:

points += 5;

break;

case 6:

points += 6;

break;

case 7:

points += 7;

break;

case 8:

points += 8;

break;

case 9:

points += 9;

break;

case 10:

points += 10;

break;

case 11:

points += 11;

break;

case 12:

points += 12;

break;

case 13:

points += 13;

break;

}

#pragma endregion

//If a minus number entered exit loop

if (cardChoice == 0)

{

//Exit Round loop

round = 'R';

//Exit Loop using R and Enter Menu using exit

exit = 'N';

}

//If Player count is over amount of Players playing

if (player >= numPlayers)

{

//Reset Count

player = 0;

//Exit Loop

round = 'R';

}

//Update Player Counter

player++;

} while (round != 'R');

//If Exit to menu selected

if(exit != 'N')

{

//Update Round Number

roundNumber++;

//End Of Round- Find Winner

#pragma region EndOfRound

#pragma region FindWinner

//Check if any players have the same card

for (int f = 0; f < numPlayers; f++)

{

for (int ff = f + 1; ff < numPlayers; ff++)

{

if (playerSelectedCard[f][f] == playerSelectedCard[ff][ff])

{

playerSelectedCard[f][f] = 0;

playerSelectedCard[ff][ff] = 0;

}

}

}

//Find Winner

for (int w = 0; w < numPlayers; w++)

{

if (playerSelectedCard[w][w] > winningCard)

{

winningCard = playerSelectedCard[w][w];

winner = (w + 1);

}

}

#pragma endregion

//If Round is a Draw

if (winner == 0)

{

if (roundNumber < 13)

{

//Display Round is a draw

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nThis Rounds was a draw!");

printf("\nPoints rolled over to next round");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

//If last round is a draw

else if (roundNumber == 13)

{

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nLast Round was a draw!");

printf("\nPoints Lost to Game!!");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

//If the round has a winner

else if (winner > 0)

{

//Display Winner or round Screen

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nThis Rounds winner is Player: %d", winner);

printf("\nTheyve scored this rounds points of %d", points);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

//Add Points to Players Score

playerScores[(winner-1)] += points;

//Reset Points

points = 0;

}

//If 13th round is over

if (roundNumber == 14)

{

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t\tEND OF GAME");

Display(playerCard, playerSuit, numPlayers, playerScores, roundNumber, playerUsedCards);

exit = 'N';

}

else

{

//Reset Round Loop

round = 'Y';

winningCard = 0;

winner = 0;

}

}

#pragma endregion

}while(exit != 'N');

//If player Enters a minus number at any point of game Menu is shown

//Menu to decide what to do

Menu(playerCard, playerSuit, numPlayers, playerScores, roundNumber, playerUsedCards);

}

//----------------------------------------------------------------------------

//Display- display Game Status

void Display(int playerCard[players][card], int playerSuit[players][suit], int numPlayers, int playerScores[players], int round, int playerUsedCards[players][card])

{

//Declare Variables

int winner = 0, winScore = 0, leave;

//Display Score List

printf("\nThe Score list is as shown:");

//Loop for Scores

for (int dis = 0; dis < numPlayers; dis++)

{

//Display Each Player and Their Scores

printf("\nPlayer %d:\t%d", (dis + 1), playerScores[dis]);

//Find Winner

if (playerScores[dis] > winScore)

{

//Set Winning Variables

winScore = playerScores[dis];

winner = (dis + 1);

}

}

//if asked for display during a game

if (round < 13)

{

//Display who is winning and the round it is

printf("\nWinner of Game at End of Round %d is: %d\n\n", (round - 1), winner);

//Continue Game

PlayGame(playerCard, playerSuit, numPlayers, playerScores, round, playerUsedCards);

}

//If it was the last round display Game Winner

else

{

//Winner of game at end

printf("\nWinner of Game is: %d\nCongratulations!!\n", winner);

printf("\n\n\nGame is Over!!\nPlease press 0 to exit or any other number to start a new game:");

scanf("%d", &leave);

if (leave <= 0)

{

exit(0);

}

else

{

NewGame();

}

}

}

//----------------------------------------------------------------------------

//New Game

void NewGame()

{

//Declare Variables

int numPlayers, cardNum, cardSuit;

#pragma region Declare Arrays

//Declare Arrays

//Card array is for numbered 1-13, repeated 4 times for suits

int cards[52] = { 1,2,3,4,5,6,7,8,9,10,11,12,13,

1,2,3,4,5,6,7,8,9,10,11,12,13,

1,2,3,4,5,6,7,8,9,10,11,12,13,

1,2,3,4,5,6,7,8,9,10,11,12,13 };

//suit array is to assign the cards to the suits

int suits[52] = { 1,1,1,1,1,1,1,1,1,1,1,1,1, //Diamonds

2,2,2,2,2,2,2,2,2,2,2,2,2, //Hearts

3,3,3,3,3,3,3,3,3,3,3,3,3, //Spades

4,4,4,4,4,4,4,4,4,4,4,4,4 }; //Clubs

//Player 1-4 array to store player cards.

int playerCard[players][card];

int playerSuit[players][suit];

//Player Used Cards

int playerUsedCards[players][card];

//Player Score Array

int playerScore[players] = { 0,0,0,0 };

#pragma endregion

//Ask how many Players

printf("Please enter number of players.Min 2 & Max 4:\t\t");

scanf("%d", &numPlayers);

//If too many players entered

if (numPlayers > 4)

{

//Display that num over Max players

printf("\nNumber entered is over MAX players allowed!!");

printf("\nNumber of players changed to MAX of 4");

//Change Number of Players to MAX

numPlayers = 4;

}

//If too little players entered

if (numPlayers < 2)

{

//Display that num over Max players

printf("\nNumber entered is under MIN players allowed!!");

printf("\nNumber of players changed to MIN of 2");

//Change Number of Players to MAX

numPlayers = 2;

}

//Reset Used Card Array

for (int play = 0; play < numPlayers; play++)

{

for (int ca = 0; ca < card; ca++)

{

//Make sure each is reset for new game

playerUsedCards[play][ca] = 0;

}

}

//Assign player cards in player array

for (int p = 0; p < numPlayers; p++)

{

//Generate Random cards

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

{

//Use Random class to pick player cards and suits

cardNum = cards[rand() % 52];

cardSuit = suits[rand() % 52];

//Assign card and suit to player

playerCard[p][i] = cardNum;

playerSuit[p][i] = cardSuit;

}

}

//Start Game

PlayGame(playerCard, playerSuit, numPlayers, playerScore, 1, playerUsedCards);

}

//----------------------------------------------------------------------------

//Continue Game- Here will read from file

void ContinueGame()

{

//Create Arrays

int playerCardLoad[players][card], playerSuitLoad[players][suit];

int playerUsedCardsLoad[players][card], playerScores[players];

//Declare Variables

int numPlayers = 0, roundNum = 0;

//Load Game details

#pragma region ReadFromFile

//Declare File Names

FILE\* gameStatus;

FILE\* gameScores;

//Read from Files

gameStatus = fopen("GameSaved.txt", "r");

gameScores = fopen("GameScore.txt", "r");

//Check if File is Open

if (gameStatus == NULL || gameScores == NULL)

{

//If Files Cant be Opened

printf("Sorry File could not be opened\n");

}

//If File can be Opened

else

{

//Check how many Players game has and round it was on

fscanf(gameStatus, "%d %d\n\n", &numPlayers, &roundNum);

//Repeat for amount of players there is

for (int p = 0; p < numPlayers; p++)

{

//Repeat 13 times to save cards and suits

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

{

//Save card details to arrays

fscanf(gameStatus, "%d %d\n", &playerCardLoad[p][i], &playerSuitLoad[p][i]);

}

//Skip line between players (as done when entering)

fscanf(gameStatus, "\n");

}

//Loop for each players used cards

for (int q = 0; q < numPlayers; q++)

{

//Only loop for 13 times- 13 cards each

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

{

//Print Card if used or not

fprintf(gameStatus, "%d\n", playerUsedCardsLoad[q][j]);

}

//Skip a line before next player

fprintf(gameStatus, "\n");

}

//Close Game status File

fclose(gameStatus);

//Game Player Scores

for (int p = 0; p < numPlayers; p++)

{

//Assign read scores to array

fscanf(gameScores, "%d\n", &playerScores[p]);

}

//Close File

fclose(gameScores);

}

#pragma endregion

//Display Game Status

Display(playerCardLoad, playerSuitLoad, numPlayers, playerScores, roundNum, playerUsedCardsLoad);

}

//----------------------------------------------------------------------------

//Menu- Select what you wish to do

void Menu(int playerCard[players][card], int playerSuit[players][suit], int numPlayers, int playerScores[players], int roundNum, int playerUsedCards[players][card])

{

//Variables

int menuChoice;

//Display Menu

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t\t\tMenu");

printf("\n\t\t Please Select an Option below");

printf("\n 1- Continue\t 2-Exit\t3-Exit & Save\t4-Display Status\nChoice:\t");

scanf("%d", &menuChoice);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//Selected Menu Option

switch (menuChoice)

{

//Continue Game

case 1:

PlayGame(playerCard, playerSuit, numPlayers, playerScores, roundNum, playerUsedCards);

break;

//Exit without saving

case 2:

exit (0);

break;

//Save Game and Exit

case 3:

SaveGame(playerCard, playerSuit, numPlayers, playerScores, roundNum, playerUsedCards);

break;

//Display Status of Game

case 4:

Display(playerCard, playerSuit, numPlayers, playerScores, roundNum, playerUsedCards);

break;

}

}