#include <stdlib.h>

#include <stdio.h>

// Franklyn Gonzalez, last edited 1/28

// Bubble Sort used to sort out scores in the second column

void opponentScore (int scores[][1]) {

int i;

int pass;

int temp;

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

printf("\n%i", scores[i][1]);

printf("\n");

for (pass = 1; pass <= 4 - 1; pass++) {

for (i = 0; i < 4 - 1; i++) {

if (scores[i][1] > scores[i + 1][1]) {

temp = scores[i][1];

scores[i][1] = scores[i + 1][1];

scores[i + 1][1] = temp;

}

}

}

}

void gamesWon(int mrPacman[], int msPacman[]) {

int win = 0, loss = 0, ties = 0;

int count = 0;

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

{

if (mrPacman[count] > msPacman[count])

{

win++;

}

else if (mrPacman[count] < msPacman[count])

{

loss++;

}

else if (mrPacman[count] == msPacman[count])

{

ties++;

}

count++;

}

printf(" %i. \n", win);

printf("Sorry.... Ms. Pacman beat you by %i games. Good luck next time!\n", loss);

}

void currentResults(int win, int loss, int ties) {

printf("Mr. Pacman wins by %i. \n He loses %i games. \n Games Tied: %i\n", win, loss, ties);

}

void currentRecord(int mrPacman[], int msPacman[]) {

int win = 0, loss = 0, ties = 0;

int count = 0;

// used as a counter to keep track of Mr and Ms Pacman

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

{

if (mrPacman[count] > msPacman[count])

{

win++;

}

else if (mrPacman[count] < msPacman[count])

{

loss++;

}

else if (mrPacman[count] == msPacman[count])

{

ties++;

}

count++;

}

currentResults(win, loss, ties);

}

// Player Two is msPacman, the opponent

int gameResultP2(int msPacman[], int y) {

printf("Ms. Pacman\n");

scanf\_s("%i", &msPacman[y]);

return msPacman[y];

}

// Player One is mrPacman, the main player

int gameResultP1(int mrPacman[], int x) {

printf("Mr. Pacman\n");

scanf\_s("%i", &mrPacman[x]);

return mrPacman[x];

}

main() {

// x is used for switch

int x;

// all variables and arrays initialized

int y = 0, z = 0;

int m = 0, n = 0;

int win = 0, loss = 0;

int mrPacman[5] = { 0 };

int msPacman[5] = { 0 };

int scores[5][2] = { { 0 } , { 0 } };

do

{

// Maximum is 5 Game Results

printf("Welcome to PACMAN vs. MS. PACMAN! Maximum 5 Game Results. \n");

printf("A) Enter game result\n");

printf("B) Current Record (# of wins and # of losses and # of ties)\n");

printf("C) Display ALL results from all games won.\n");

printf("D) Display ALL results ordered by opponent score from low to high.\n");

printf("E) Quit\n");

while ((getchar()) != '\n');

x = getchar();

switch (x)

{

case 'A':

// 2D array gets input from gameResults

scores[y][0] = gameResultP1(mrPacman, m++);

scores[y][1] = gameResultP2(msPacman, n++);

y++;

break;

case 'B':

// Calls currentRecord and prints games between Mr. Pacman and Ms. Pacman

currentRecord(mrPacman, msPacman);

printf("Current record displayed!\n");

break;

case 'C':

// Calls gamesWon and prints all games won by Mr. Pacman

printf("Displaying all results from all games won by Mr. Pacman...");

gamesWon(mrPacman, msPacman);

break;

case 'D':

// Displays opponent's score, Ms. Pacman, from lowest to highest.

printf("Displaying all results by opponent score from low to high...");

opponentScore(scores);

break;

case 'E':

printf("Exiting game program.");

break;

default:

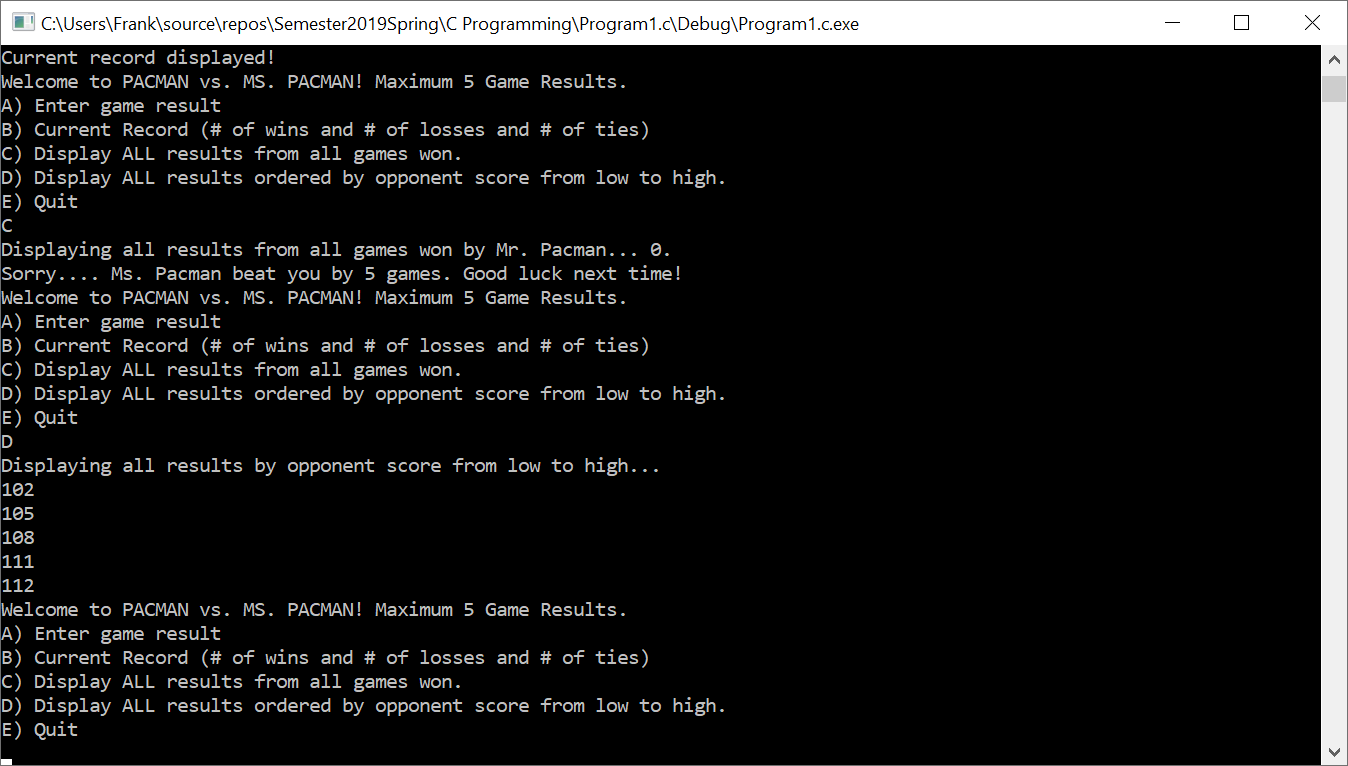
break;

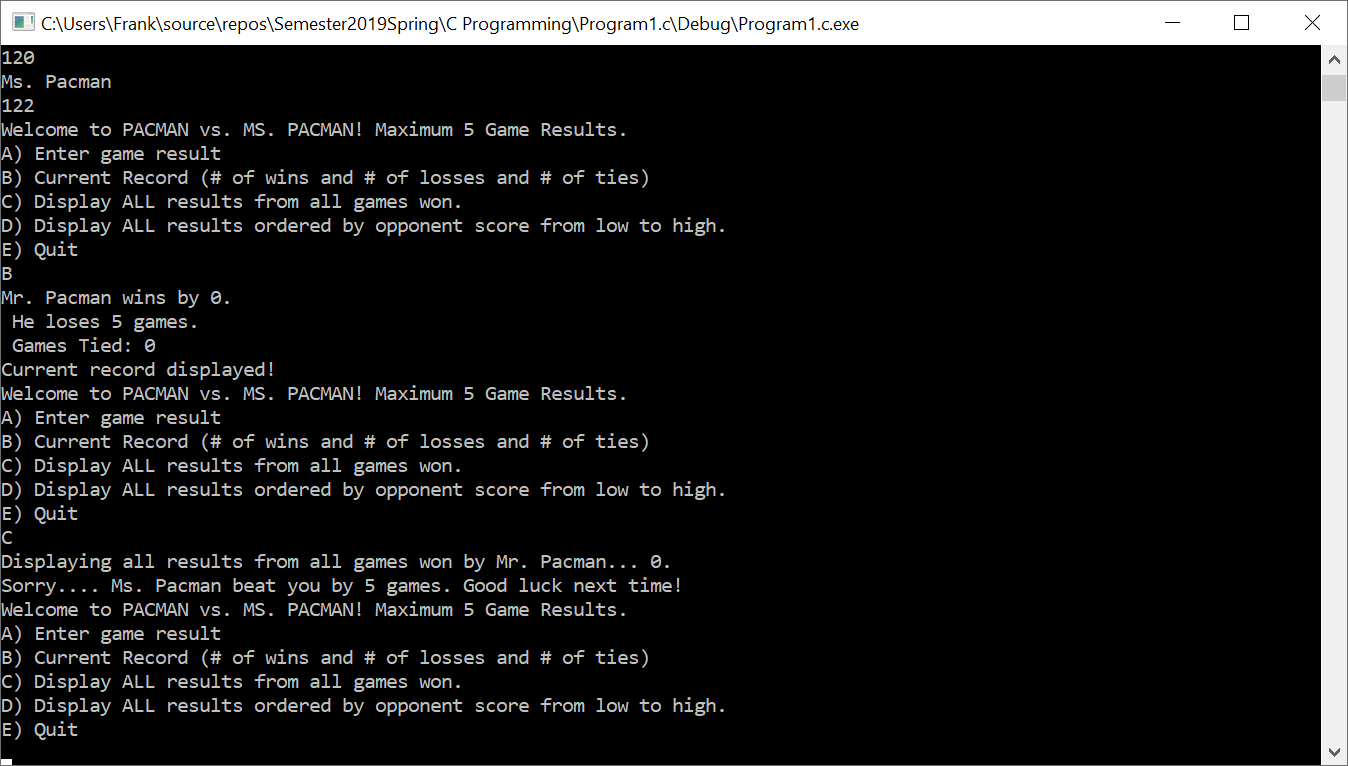
}

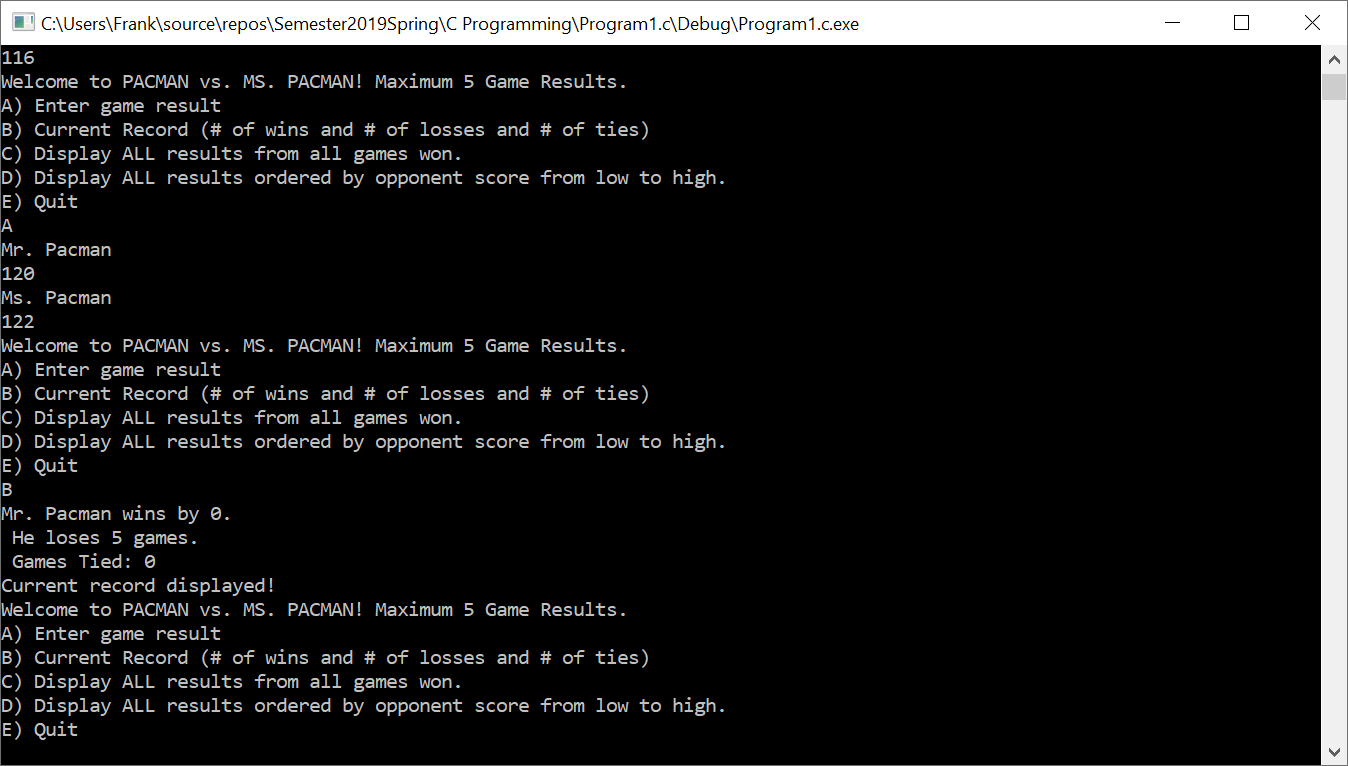
} while (x == 'A' || x == 'B' || x == 'C' || x == 'D' || x != 'E' );

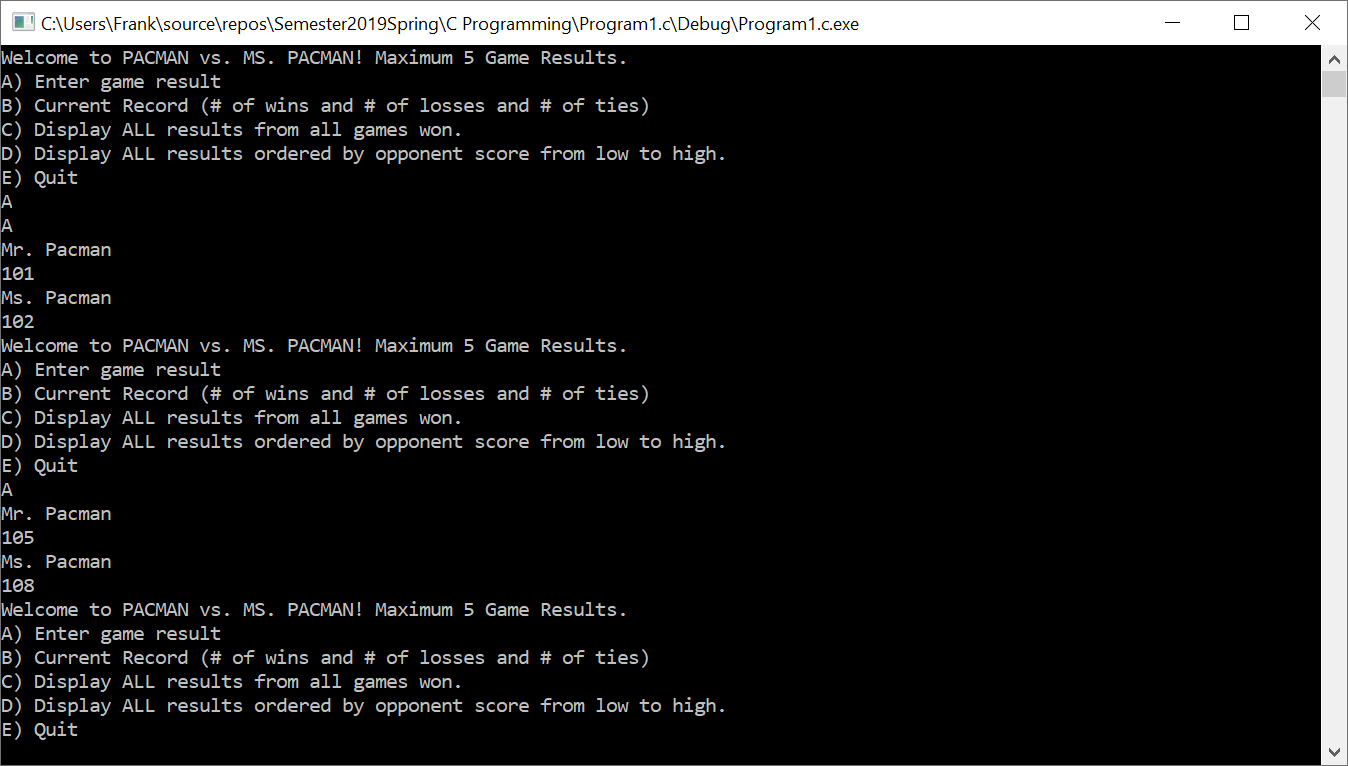
return 0;

}









|  |  |  |
| --- | --- | --- |
| **Input**  x  y  z  m  n  win  loss  mrPacman[5]  msPacman[5]  scores[5][2]  gameResultP1  gameResultP2 | **Processing**  🡪 Takes in user’s character🡪  🡪 Used as a counter for scores 2d array🡪  z  m  n  🡪 Used to calculate wins for Player 1, Mr. Pacman🡪  🡪Used to calculate losses from Player 1🡪  🡪 Stored 5 game results into the memory🡪  🡪 Stored 5 game results into the memory🡪  🡪 Used to store scores from both players 🡪  🡪 Used to input Mr. Pacman’s score🡪  🡪 Used to input Ms. Pacman’s score🡪 | **Output**  A,B,C,D,E, default  Adds 1 to y  Adds to mrPacman’s array  Adds to msPacman’s array  Prints wins by P1  Prints losses by P1  Prints P1’s results  Prints P2’s results  Separates scores by P1 & P2  Outputs P1’s score  Outputs P2’s score |