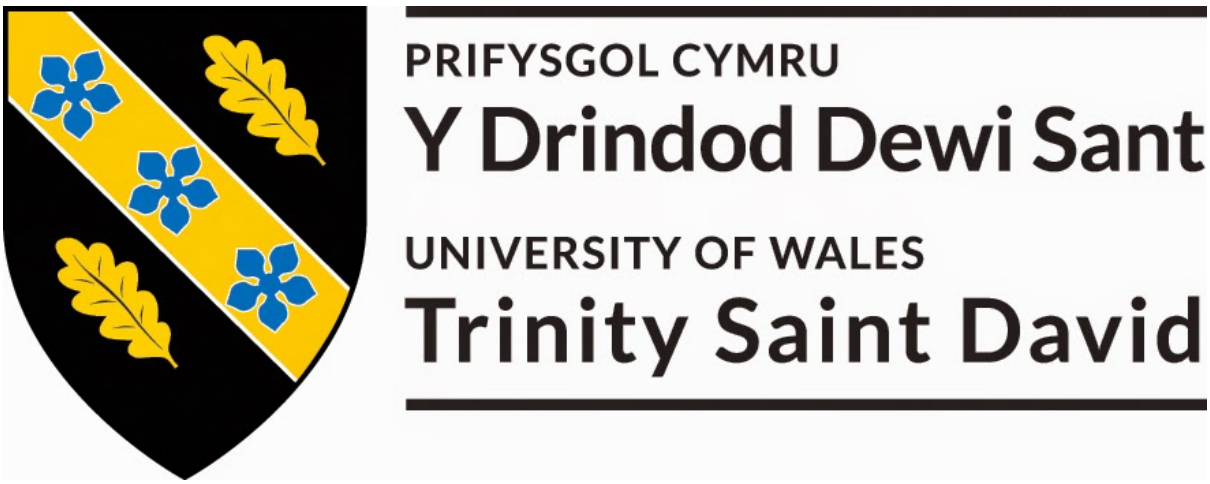
Programs in C and Documentation



Siobhan Kelly Stephens

1800709

1st May 2019

Software Development

School of Applied Computing

Table of Contents

1. Pairs Game 4

1.1. Problem 4

1.1. Analysis of Solution 5

1.1.1. Problem Break Down 5

1.1.2. Pseudocode 6

1.2. Annotated C Code Listing 9

1.3. Testing 19

1.3.1. Test Plan 19

1.3.2. Test Data 21

1.3.3. Test Runs and Results 23

1.4. Limitations of the Solution 50

Part C

1. Development and testing completed using the IDE Xcode (on Mac OSX 10.14)

2. Testing evidence consists of screenshots of the input/output from the CLI in Xcode IDE

1. Pairs Game

Develop a C program to allow the user to play a version of a popular card matching game.

1	1	2	2
3	3	4	4
5	5	6	6

Figure 1. Example of Card Matching Game Deck (sorted)

1.1. Problem

The game aim of the pairs game is to find all six pairs of numbers (1-6) in the smallest number of attempts (minimum score/attempts is six).

The pairs game must allow a user to pick two cards and view their contents, to check for a match. A running top-ten leader board must be kept using a text file, which will contain user forenames and scores. Once all six pairs have been matched the score will compared with the up-to-date text file, and if found to be within the top-10 scores, the user's name and score will be saved to the file, only the best ten scores will be kept in the file at any given time.

Therefore, the program must allow the user to input their forename and allow them to enter a card number which represents the card they wish to turn over. A card may not be turned over again once it has been matched, therefore validation must be put in place to prevent this. Further validation to only allow the user to select card numbers one to twelve must also be implemented.

1.1. Analysis of Solution

1.1.1. Problem Break Down

a. Ask user to input their name
b. Ask user if they want to play and capture their answer
c. If the answer is yes: Play game
    a. Print the rules
    b. Print the empty card deck
    c. Loop while game is not over
        i. Ask the user for the first card they want to turn
        ii. Ask the user for the second card they want to turn
        iii. Validate both cards (Must not be same two cards, must be between 1 and 12, must not be a card that has already been matched)
        iv. Print the card deck with the two turned card values
        v. If the cards match:
            1. Set the cards to be equal to 99 (so they can't be picked again)
        vi. Increment the score counter
        vii. Increment the attempt counter
        viii. Check how many pairs have been matched
        ix. If all pairs have been matched
            1. Set to be game over
            2. Read the current top-10 score file to two arrays, one for name and another for score
            3. Amend the current user's name and score to the arrays
            4. Sort the array in ascending order
            5. Write the top 10 scores to the file
d. If the answer is no:
    a. End
e. Else:
    a. Ask the user again if they want to play

1.1.2. Pseudocode

Declare global integer array

card_deck[12] = {1,1,2,2,3,3,4,4,5,5,6,6}

Main Function

```
Declare variables
Print (welcome message)
Print (please enter your first name)
Input (first_name)
Print (do you want to start?)
Input (answer)
While (answer is not no):
        if (answer is yes):
                play_card_matching_game() *see below
        else if (answer is no):
                Print (Goodbye)
        else:
                Print (Sorry, please try entering that again)
        Print (Do you want to play again?)
        Input (answer)
End while
```

Play Game Function

```
Declare variables
Print (Rules and info)
Print (Empty card deck) *simple graphical representation
generate_random_card_deck()
While (not game over):
        matches = matches + pick_cards()
        Print (current number of matches: + matches)
```

```
        increment attempt counter
        Print (your current attempt is: + attempt_counter)
        Check how many pairs have been matched
        If (pairs_matched = 6):
                Print (Congratulations. All cards matched)
                Set game_over = 1
                Print (final score is: + attempt_counter)
                Read score file
                Add current user score and user forename to list
                Bubble sort list
                Write to top-10 to score file
        End if
```

Int Pick Cards Function

```
Print (Please pick the first card you want to turn over (1-12))
Input (card_1)
While (card_1 is not valid):
        Print (Sorry, please pick a card between 1 and 12)
        Input (card_1)
End while
While (card_1 has been matched):
        Print (Sorry, that card has already been matched, pick another)
        Input (card_1)
End while
Print (Please pick the second card you want to turn over (1-12))
Input (card_2)
While (card_2 is not valid):
        Print (Sorry, please pick a card between 1 and 12)
        Input (card_2)
End while
While (card_2 has been matched):
        Print (Sorry, that card has already been matched, pick another)
        Input (card_2)
End while
score = check_cards_are_match(card_1, card_2)
```

```
If (score > 0):
        Print (You've found a pair)
        Set card_1 = 99
        Set card_2 = 99
End if
Return score
```

Generate Random Card Deck Function

```
Declare variables
For loop (100 iterations):
random_index_1 = random number between 0 and 11
random_index_2 = random number between 0 and 11
temp = card_deck[random_index_1]
card_deck[random_index_1] = card_deck[random_index_2]
card_deck[random_index_2] = temp
End for
```

1.2. Annotated C Code Listing

```c
//includes are necessary for importing libraries
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>

//declare global integer array to store card deck values
int card_deck[12] = {1,1,2,2,3,3,4,4,5,5,6,6};

//declare procedure/function prototypes
void print_empty_deck(void);
void print_card(int,int);
int pick_cards(void);
void print_name(char*, int);
void read_sort_write_scores_file(char*, int, int);
void generate_random_card_deck(void);
int pick_cards(void);
void play_card_matching_game(char*, int);
int check_card_match(int,int);
int validate_card_number(int);
int validate_selected_card(int);
void play_card_matching_game(char*,int);


/* Print empty deck is used to provide the user with a simple graphical
representation
 * of the 4 x 3 card deck.
 */
void print_empty_deck(){
    printf("     ----       ----       ----       ---- \n");
    printf("  1.|    |   2.|    |   3.|    |   4.|    | \n");
    printf("     ----       ----       ----       ---- \n");
    printf("     ----       ----       ----       ---- \n");
    printf("  5.|    |   6.|    |   7.|    |   8.|    | \n");
    printf("     ----       ----       ----       ---- \n");
    printf("     ----       ----       ----       ---- \n");
    printf("  9.|    |  10.|    |  11.|    |  12.|    | \n");
    printf("     ----       ----       ----       ---- \n");
}

/* Print card function takes two integers as parameters.
 * It's function is to print a graphical representation of the 4 x 3 card
deck.
 * It uses switch statements to set the graphical card values to the
values in the card_deck array.
*/
void print_card(int card_index_1, int card_index_2){
    int card_value_1 = 0; int card_value_2 = 0; int card_value_3 = 0; int
card_value_4 = 0;
    int card_value_5 = 0; int card_value_6 = 0; int card_value_7 = 0; int
card_value_8 = 0;
    int card_value_9 = 0; int card_value_10 = 0; int card_value_11 = 0;
int card_value_12 = 0;
    switch (card_index_1){
        case 1:
```

```
            card_value_1 = card_deck[0];
            break;
        case 2:
            card_value_2 = card_deck[1];
            break;
        case 3:
            card_value_3 = card_deck[2];
            break;
        case 4:
            card_value_4 = card_deck[3];
            break;
        case 5:
            card_value_5 = card_deck[4];
            break;
        case 6:
            card_value_6 = card_deck[5];
            break;
        case 7:
            card_value_7 = card_deck[6];
            break;
        case 8:
            card_value_8 = card_deck[7];
            break;
        case 9:
            card_value_9 = card_deck[8];
            break;
        case 10:
            card_value_10 = card_deck[9];
            break;
        case 11:
            card_value_11 = card_deck[10];
            break;
        case 12:
            card_value_12 = card_deck[11];
            break;
    }
    switch (card_index_2){
        case 1:
            card_value_1 = card_deck[0];
            break;
        case 2:
            card_value_2 = card_deck[1];
            break;
        case 3:
            card_value_3 = card_deck[2];
            break;
        case 4:
            card_value_4 = card_deck[3];
            break;
        case 5:
            card_value_5 = card_deck[4];
            break;
        case 6:
            card_value_6 = card_deck[5];
            break;
        case 7:
            card_value_7 = card_deck[6];
            break;
        case 8:
```

```
            card_value_8 = card_deck[7];
            break;
        case 9:
            card_value_9 = card_deck[8];
            break;
        case 10:
            card_value_10 = card_deck[9];
            break;
        case 11:
            card_value_11 = card_deck[10];
            break;
        case 12:
            card_value_12 = card_deck[11];
            break;
    }
    //set card values to be 99 (Meaning that the card has already been
matched)
    if(card_deck[0] == 99){
        card_value_1 = 99;
    }
    if(card_deck[1] == 99){
        card_value_2 = 99;
    }
    if(card_deck[2] == 99){
        card_value_3 = 99;
    }
    if(card_deck[3] == 99){
        card_value_4 = 99;
    }
    if(card_deck[4] == 99){
        card_value_5 = 99;
    }
    if(card_deck[5] == 99){
        card_value_6 = 99;
    }
    if(card_deck[6] == 99){
        card_value_7 = 99;
    }
    if(card_deck[7] == 99){
        card_value_8 = 99;
    }
    if(card_deck[8] == 99){
        card_value_9 = 99;
    }
    if(card_deck[9] == 99){
        card_value_10 = 99;
    }
    if(card_deck[10] == 99){
        card_value_11 = 99;
    }
    if(card_deck[11] == 99){
        card_value_12 = 99;
    }
    //Print the graphical representation of the 4 x 3 card deck
    //%02d is used to add a leading 0 to a single digit number, so that
the graphical spacing will
    //appear correct whether the number is 1 or 11 (displayed as 01 or
11).
    printf("     ----       ----       ----       ---- \n");
```

```
    printf("  1.| %02d |   2.| %02d |   3.| %02d |   4.| %02d |
\n",card_value_1, card_value_2, card_value_3, card_value_4);
    printf("     ----       ----       ----       ---- \n");
    printf("     ----       ----       ----       ---- \n");
    printf("  5.| %02d |   6.| %02d |   7.| %02d |   8.| %02d |
\n",card_value_5, card_value_6, card_value_7, card_value_8);
    printf("     ----       ----       ----       ---- \n");
    printf("     ----       ----       ----       ---- \n");
    printf("  9.| %02d |  10.| %02d |  11.| %02d |  12.| %02d |
\n",card_value_9, card_value_10, card_value_11, card_value_12);
    printf("     ----       ----       ----       ---- \n");

}
/* Print name procedure takes two parameters:
 * 1. The pointer to the memory location of the forename array position 0
 * 2. The length of the name
 * This function is used only for testing purposes.
 */
void print_name(char *ptr_name, int forename_len){
    printf("printing name... \n");
    for (int i = 0; i < forename_len; i++){
        //check that the memory location points to an alpha character
(don't print overflow)
        if(isalpha(*ptr_name)){
            //if it is, print out that character
            printf("%c",*ptr_name);
        }
        ptr_name ++;
    }
    printf("\nfinished printing... \n");
}

/* Read/Sort/Write scores file takes three parameters:
 * 1. The pointer to the memory location of the array index 0 that is used
to store the user's forename
 * 2. The length of the user's forename (used for looping to avoid
overflow)
 * 3. The user's score from the game just finished
 * It firstly reads the current Top 10 score file, which stores 10 records
only, and stores the names and scores into an array
 * It then appends the current user's score and forename to both arrays
 * Then, the scores are sorted using a bubble sort.
 * Next, the score board is printed for the user to see.
 * Finally, the top 10 records are written to the existing file
(overwrites)
*/
void read_sort_write_scores_file(char *ptr_name, int forename_len, int
user_score){
    //declare variables
    int i = 0; int j = 0;
    char name[11][20];
    int score[11];
    int temp_no;
    char temp_names[100][20];

    //**** File will always only contain the top 10 records ****
    FILE *fp;

    fp = fopen("pairs_score_top10.txt", "r");//open file
```

```
    //read file names and scores to arrays name and score
    //loop 10 times for the 10 records stored in the file.
    for (i = 0; i < 10; i++){
        fscanf(fp, " %s %d", &name[i], &score[i]);
//        printf("* name: %s, score: %d\n",name[i], score[i]); //printf
for testing purposes
    }
    fclose(fp); //close file

    //amend the user's score and name to the two arrays
    //loop through using the forename length as the loop count
    for (i = 0; i < forename_len; i++){
        //check that the memory location points to an alpha character
(don't add overflow)
        if(isalpha(*ptr_name)){
            //if it is, print out that character (testing)
            //printf("%c",*ptr_name);
            //set name array[10] to char at memory location
            name[10][i] = *ptr_name;
        }
        ptr_name ++;//increment the memory location pointer
    }
    score[10] = user_score;//amend user's score to the score array

    //bubble sort the scores before writing the top 10 to the file
    for (i = 0; i < 11; i++){
        for (j = 0; j < 11; j++){
            if(score[j] > score[j+1]){//sort ascending
                //swap int scores using temp variable
                temp_no = score[j];
                score[j] = score[j+1];
                score[j+1] = temp_no;
                //swap names using strcpy()
                strcpy(temp_names[j], name[j]);
                strcpy(name[j], name[j+1]);
                strcpy(name[j+1], temp_names[j]);
            }
        }
    }
    //print the current top 10 leaderboard (user's score may or may not be
in leaderboard)
    printf("\nCurrent Top 10 Leaderboard:\n");
    printf("----------------------------\n");
    //print scores using for loop (10 iterations)
    for (i = 1; i < 11; i++){
        printf(" %02d | %s\n",score[i],name[i]);
    }
    printf("----------------------------\n");
    //write top 10 to the file by looping through the top 10 sorted
records
    fp = fopen("pairs_score_top10.txt", "w");//2nd param is r/w read/write
    for(i = 1; i < 11; i++){
        fprintf(fp, "%s %d\n" , name[i],score[i]);
    }
    fclose(fp); //close file

}

/* Generate random card deck is used to shuffle the global card_deck array
```

```
 * Whilst in a for loop (100 iterations), it selects two random indexes
from the array (0-11),
 * and then swaps them. It does this 100 times so that the card deck
should be different
 * each time the user plays the game.
 */
void generate_random_card_deck(void){
    //declare variables
    int temp = 0; int max = 11; int min = 0;
    int rand_index_1 = 0; int rand_index_2 = 0;

    //loop 100 times
    for(int i = 0; i < 100; i++){
        rand_index_1 = (rand() % (max + 1 - min)) + min;
        rand_index_2 = (rand() % (max + 1 - min)) + min;
        temp = card_deck[rand_index_1];
        card_deck[rand_index_1] = card_deck[rand_index_2];
        card_deck[rand_index_2] = temp;
    }
    //print the shuffled deck using a for-loop. Used for testing purposes
only.
    //printf("\n\n** shuffled deck: \n");
//    for(int y = 0; y < 12; y++){
//        printf("* %d ",card_deck[y]);
//    }
//    printf("\n");
}

/* Pick cards is used to ask the user to input two integers to represent
 * the two cards they want to turn over.
 * It uses while loops to continually ask the user for input while the
cards
 * they pick are invalid (uses functions validate_card_number and
 * validate_selected_card).
 * Once the two cards are valid, it checks to see if the values are a
match.
 * If they are a match, it sets the card values to be 99 so they cannot be
matched again.
 * It returns an integer score to the calling function.
 */
int pick_cards(void){
    int card_1 = 0; int card_2 = 0; int score = 0;
//    printf("\n\n** Current Card Deck: \n**");
//    for(int y = 0; y < 12; y++){
//        printf("* %d ",card_deck[y]);
//    }
//    printf("\n");
    printf("\nPlease pick the card you want to turn over (1-12)\n");
    scanf("%d",&card_1);//store value as card_1
    //while the card isn't valid, keep asking the user for additional
input
    while(validate_card_number(card_1) == 0){
        printf("Sorry, wrong card number. Please enter the card you want
to turn over (1-12).\n");
        scanf("%d",&card_1);
    }
    while(validate_selected_card(card_1) == 0){
        printf("Sorry, that card has already been matched. Please enter
the card you want to turn over (1-12).\n");
```

```
        scanf("%d",&card_1);
    }
    printf("\nPlease pick the next card you want to turn over (1-12)\n");
    scanf("%d",&card_2);
    while(validate_card_number(card_2) == 0){
        printf("Sorry, wrong card number. Please enter the next card you
want to turn over (1-12).\n");
        scanf("%d",&card_2);
    }
    while(validate_selected_card(card_2) == 0){
        printf("Sorry, that card has already been matched. Please enter
the next card you want to turn over (1-12).\n");
        scanf("%d",&card_2);
    }
    if(card_1 == card_2){
        printf("You can only turn over one card at once! Please pick the
next card you want to turn over (1-12).");
        scanf("%d",&card_2);
    }
    print_card(card_1,card_2);

    //check if the cards match
    score = check_card_match(card_deck[card_1-1], card_deck[card_2-1]);
    //reset deck so those cards cannot be picked again
    if(score > 0){
        printf("\n!* Well done, you found a pair of %d's!!
*!\n\n",card_deck[card_1-1]);
        card_deck[card_1-1] = 99;
        card_deck[card_2-1] = 99;

    }
    return score;
}

/* Play card matching game takes 2 parameters.
 * 1. The pointer to the memory location of the array index 0 storing the
user's forename
 * 2. The length of the user's forename (avoid overflow)
 * The purpose of the function is to manage the card matching game.
 * From here many other functions are called.
*/
void play_card_matching_game(char *ptr_name, int forename_len){
    //declare variables
    int matches = 0; int attempt_counter = 0;
    int game_over = 0; int match_count = 0;

    //call generate random card deck function to shuffle the card deck
    generate_random_card_deck();
    //print out info for user
    printf("\nTo start, I will ask you to pick a card. The deck is
numbered 1 to 12.\n");
    printf("\nI will then ask you to pick another card.\n");
    printf("\nIf they match, the cards will be removed from the deck and
represented using '99'.\n Otherwise, I wil ask you for another
card.\n\n");
    printf("\nReady?\n\n");

    //print the empty card deck
    print_empty_deck();
```

```
    //loop while the game isn't over
    while(game_over == 0){
        matches = matches + pick_cards();//set matches to be equal to
matches plus return of function
        printf("\nYour current number of matches is: (%d)\n",matches);
        attempt_counter++;//increment the attempt counter
        printf("Your current attempt count is: (%d)\n",attempt_counter);
        match_count = 0;
        //check if the game is over and all pairs have been matched
        for(int i = 0; i < 12; i++){
            //if the card deck value is 99, the card has been matched (set
in previous function)
            if(card_deck[i] == 99){
                match_count++;
            }
        }

//        printf("**match count: %d\n",match_count); //used for testing
        //if all the pairs have been matched
        if(match_count==12){
            printf("\n*** !! Congratulations, you've matched all the
cards!! ***\n");
            game_over = 1; //set variable game over to be 1 (ends the
while loop)
            printf("\nYour final score is: %02d\n",attempt_counter);
            //call the function that reads the top-10 score,
            //amends the user's name and score to the list,
            //sorts the scores and writes the top-10 back to the file

read_sort_write_scores_file(ptr_name,forename_len,attempt_counter);
        }
    }
}
/* Check card match takes two parameters, card_1_value and card_2_value.
 * If the two card values match, TRUE (or 1) is returned. Else False/0.
 */
int check_card_match(int card_1_value, int card_2_value){
    if(card_1_value == card_2_value){
        return 1;
    }
    return 0;
}
/* Validate card number is used to check that the card number (parameter),
 * is a valid card number. The criteria is that the card number must be
 * between 1 and 12. If it is valid, true is returned, else false.
 */
int validate_card_number(int card_number){
    if (card_number < 1 || card_number > 12){
        return 0;
    }
    return 1;
}
/* Validate selected card takes a single integer parameter, which it uses
 * to check if the value of that card is equal to 99.
 * If the value is equal to 99, then the card has already been matched.
 * If it has been matched, FALSE is returned, otherwise, TRUE is returned.
 */
int validate_selected_card(int selected_card){
```

```
    if(card_deck[selected_card-1]==99){
        return 0;
    }
    return 1;
}


int main(int argc, const char * argv[]) {
    srand(time(NULL)); //seed for randomise
    int play_game = 2;//play game is initialised to 2 to begin while loop.
    int forename_len = 0;//forename_len is used to avoid overflow from
pointer
    char *ptr_forename;//declare pointer for forename
    int forename_val = 1;
    char forename[20];//declare array for forename
    ptr_forename = &forename[0];//point pointer to memory location of
forename array[0]

    printf("\n\n- - - Hello, Welcome to the Card Matching Game - - -
\n\n");
    printf("Please enter your first name:\n");
    scanf(" %s",&forename);//store input as forename
    forename_len = strlen(forename);//set variable to length of forename
using strlen function

    //validate forename length and check is it contains only alpha
characters
    for (int i=0; forename[i]!= '\0'; i++){
        // check for alphabets
        if (isdigit(forename[i]) != 0){
            forename_val = 0;
        }
    }
    //loop abd ask user for new input while it is invalid
    while(forename_len<2 || forename_len>20 || forename_val == 0){
        forename_val = 1; //reset boolean for comparison
        printf("Sorry, try again. Please enter your first name:\n");
        scanf(" %s",&forename);//store input as forename
        //loop through the forename to check for digits
        for (int i=0; forename[i]!= '\0'; i++){
            if(isalpha(forename[i]) !=1 ){
                forename_val = 0;//if it is not alpha, set to be false
            }
            if (isdigit(forename[i]) != 0){
                forename_val = 0;//if it does contain a digit, set to be
false
            }
        }
        forename_len = strlen(forename);//set variable to length of
forename using strlen function
    }

    //print name used to test that the pointer is being transferred
correctly
//    print_name(ptr_forename,forename_len);

    printf("\n\nThanks %s. Do you want to start? Please press 1 for Yes
and 0 for No.\n",forename);
    scanf("%d",&play_game);
```

```
    //loop while the user does not want to play
    while (play_game != 0){
        printf("\n- - - - - - - - - - - - - - -\n");
        //if the answer is yes, they want to play, call the play game
function.
        if(play_game == 1){
            play_card_matching_game(ptr_forename, forename_len);
        }else if(play_game == 0){
        //if they do not want to play, print a goodbye message.
            printf("Ok. Goodbye.\n");
        }else{
            //if the input is neither 1 or 0, ask for input again.
            printf("Sorry, I didn't get that. Please try again.\n\n");
        }
        printf("\n\nDo you want to play again? Please press 1 for Yes and
0 for No.\n");
        scanf("%d",&play_game);
    }
    if(play_game == 0){
        //if they do not want to play, print a goodbye message.
        printf("Ok. Goodbye.\n");
    }
    return 0;

}
```

1.3. Testing

1.3.1. Test Plan

The test plan will outline tests to be performed on the system to ensure the system accepts valid inputs and produces the expected output. These will test against three main types of data: valid, invalid, and erroneous, plus both valid and invalid boundary data.

TEST #	TEST	PURPOSE	TEST DATA	EXPECTED RESULT
		Tests For User Inputs		
I.1	Enter valid data for **Forename**	Ensure the acceptance of valid data	I.A	Pass
I.2	Enter invalid data for **Forename** (too long)	Ensure system can handle invalid data	I.B	Pass, prompt user to enter new data
I.3	Enter invalid data for **Forename** (too short)	Ensure system can handle invalid data	I.C	Pass, prompt user to enter new data
I.4	Enter invalid data for **Forename** (numeric)	Ensure system can handle invalid data	I.D	Pass, prompt user to enter new data
I.5	Enter erroneous data for **Forename**	Ensure system does not accept erroneous data	I.E	Fail (out of system scope to handle)
I.6	Enter valid boundary data for **Forename** (length)	Ensure the acceptance of valid data	I.F	Pass
I.7	Enter valid boundary data for **Forename** (length)	Ensure the acceptance of valid data	I.G	Pass
I.8	Enter valid data for **Play Game (1)**	Ensure the acceptance of valid data	I.H	Pass, play game is triggered
I.9	Enter valid data for **Play Game (0)**	Ensure the acceptance of valid data	I.I	Pass, program ends with message
I.10	Enter invalid data for **Play Game** (3)	Ensure system can handle invalid data	I.J	Pass, prompt user for new input
I.11	Enter erroneous data for **Play Game** (A)	Ensure system does not accept erroneous data	I.K	Fail (out of system scope to handle)
I.12	Enter erroneous data for **Play Game** (&^)	Ensure system does not accept erroneous data	I.L	Fail (out of system scope to handle)

TEST #	TEST	PURPOSE	EXPECTED RESULT
PO.1	If user chooses to play, ensure rules and welcome message are displayed	Ensure the system is user friendly and it is clear to the user what they must to	The rules and welcome message are displayed to the user
PO.2	If the user chooses not to play, ensure the goodbye message is shown and the program is ended	Ensure the system does what the user intended	The goodbye message is displayed and the program ends execution
PO.3	Ensure that the generate random card deck function works to effectively shuffle the deck after many iterations	Ensure the game remains interesting by effectively shuffling the deck each time the game is played	The deck is shuffled differently on each iteration of the game
PO.4	Ensure that the graphical card deck displays the correct card value based on the card deck array	Ensure that the game produces correct output so that the user can play the game	The correct card values are displayed on the graphical card deck
PO.5	Ensure that the graphical card deck displays the correct values for cards that have already been matched (99)	Ensure that it is clear to the user which cards have already been matched	The cards that have been matched show the value 99.
PO.6	Ensure that the leader board (read from text file) is outputted correctly with the correct values	Ensure that the program reads the text file correctly and outputs the values to the user	The scores are read correctly and outputted to the user in sorted order
PO.7	Ensure that the leader board is sorted correctly before being written to the file	Ensure that the bubble sort works correctly to sort the scores	The scores are sorted in ascending order
PO.8	Ensure that the leader board (written to text file) is saved correctly with the correct values (only top 10 scores)	Ensure that the leader board is written to the file correctly, so the scores are correct for the next user	The sorted scores are written to the file in order (10 scores only)

1.3.2. Test Data

TEST DATA I.A FOR TEST I.1			
Field	**Value**	**Type**	**Type of Data**
Forename	Charlene	String (char array)	Valid

TEST DATA I.B FOR TEST I.2			
Field	**Value**	**Type**	**Type of Data**
Forename	abcdefghijklmnopqrstuvwyxz	String (char array)	Invalid

TEST DATA I.C FOR TEST I.3			
Field	**Value**	**Type**	**Type of Data**
Forename	A	String (char array)	Invalid

TEST DATA I.D FOR TEST I.4			
Field	**Value**	**Type**	**Type of Data**
Forename	J1mm3	String (char array)	Invalid

TEST DATA I.E FOR TEST I.5			
Field	**Value**	**Type**	**Type of Data**
Forename	%&+	String (char array)	Erroneous

TEST DATA I.F FOR TEST I.6			
Field	**Value**	**Type**	**Type of Data**
Forename	Ab (Length 2)	String (char array)	Valid Boundary

TEST DATA I.G FOR TEST I.7			
Field	**Value**	**Type**	**Type of Data**
Forename	Abcdefghijklmnopqrst (Length 20)	String (char array)	Valid Boundary

TEST DATA I.H FOR TEST I.8			
Field	**Value**	**Type**	**Type of Data**
Play_game	1	Integer	Valid

TEST DATA I.I FOR TEST I.9			
Field	**Value**	**Type**	**Type of Data**
Play_game	0	Integer	Valid

TEST DATA I.J FOR TEST I.10			
Field	**Value**	**Type**	**Type of Data**
Play_game	3	Integer	Invalid

TEST DATA I.K FOR TEST I.11			
Field	**Value**	**Type**	**Type of Data**
Play_game	A	Integer	Invalid

TEST DATA I.L FOR TEST I.12			
Field	**Value**	**Type**	**Type of Data**
Play_game	&^	Integer	Erroneous

1.3.3. Test Runs and Results

Test I.1

Screenshot Evidence from IDE:

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Charlene


Thanks Charlene. Do you want to start? Please press 1 for Yes and 0 for No.
```

Figure 2. Input and Output / Result for Test I.1

Result:

Pass: The program accepted valid data as expected, and execution continued as expected.

Test I.2

Screenshot Evidence from IDE:

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
abcdefghijklmnopqrstuvwyxz
Sorry, try again. Please enter your first name:
abcdefghijklmnopqrstuvwyxz
Sorry, try again. Please enter your first name:
```

Figure 3. Input and Output / Result for Test I.2

Result:

Pass: The program did not accept the invalid data as expected, as it prompted the user for new input. The invalid data was entered once more, and again, it was not accepted and the user was prompted for new input.

Test I.3

Screenshot Evidence from IDE:

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
a
Sorry, try again. Please enter your first name:
a
Sorry, try again. Please enter your first name:
```

Figure 4. Input and Output / Result for Test I.3

Result:

Pass: The program did not accept the invalid data as expected, as it prompted the user for new input. The invalid data was entered once more, and again, it was not accepted and the user was prompted for new input.

Test I.4

Screenshot Evidence from IDE:

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
J1mme
Sorry, try again. Please enter your first name:
J1mme
Sorry, try again. Please enter your first name:
```

Figure 5. Input and Output / Result for Test I.4

Result:

Pass: The program did not accept the invalid data as expected, as it prompted the user for new input. The invalid data was entered once more, and again, it was not accepted, and the user was prompted for new input.

Test I.5

Screenshot Evidence from IDE:

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
%&+


Thanks %&+. Do you want to start? Please press 1 for Yes and 0 for No.
```

Figure 6. Input and Output / Result for Test I.5

Result:

Fail: The program did accept the erroneous data. In order to improve this in the future, further validation should be put in place to ensure the user may only enter characters from A-Z.

Test I.6
Screenshot Evidence from IDE:

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
ab


Thanks ab. Do you want to start? Please press 1 for Yes and 0 for No.
```

Figure 7. Input and Output / Result for Test I.6

Result:

Pass: The program accepted the valid boundary data as expected, and execution continued as expected.

Test I.7

Screenshot Evidence from IDE:

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
abcdefghijklmnopqrst


Thanks abcdefghijklmnopqrst. Do you want to start? Please press 1 for Yes and 0 for No.
```

Figure 8. Input and Output / Result for Test I.7

Result:

Pass: The program accepted the valid boundary data as expected, and execution continued as expected.

Test I.8

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
1

- - - - - - - - - - - - - - -


** shuffled deck:
* 3 * 2 * 3 * 5 * 4 * 2 * 1 * 6 * 5 * 4 * 1 * 6

To start, I will ask you to pick a card. The deck is numbered 1 to 12.

I will then ask you to pick another card.

If they match, the cards will be removed from the deck and represented using '99'.
 Otherwise, I wil ask you for another card.


Ready?

    ----       ----       ----       ----
 1.|    |   2.|    |   3.|    |   4.|    |
    ----       ----       ----       ----
    ----       ----       ----       ----
 5.|    |   6.|    |   7.|    |   8.|    |

All Output
```

Figure 9. Input and Output / Result for Test I.8

Result:

Pass: The program accepted the valid data as expected, and execution continued as expected, as the card matching game function was called.

Test I.9

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
0
Ok. Goodbye.
Program ended with exit code: 0
```

Figure 10. Input and Output / Result for Test I.9

Result:

Pass: The program accepted the valid data as expected, the goodbye message was printed and the program ended.

Test I.10

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
3

- - - - - - - - - - - - - - -
Sorry, I didn't get that. Please try again.



Do you want to play again? Please press 1 for Yes and 0 for No.
```

Figure 11. Input and Output / Result for Test I.10

Result:

Pass: The program did not accept the invalid data, and prompted the user for new input as expected.

Test I.11

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
A


- - - - - - - - - - - - - - -
Sorry, I didn't get that. Please try again.



Do you want to play again? Please press 1 for Yes and 0 for No.

- - - - - - - - - - - - - - -
Sorry, I didn't get that. Please try again.



Do you want to play again? Please press 1 for Yes and 0 for No.

- - - - - - - - - - - - - - -
Sorry, I didn't get that. Please try again.



Do you want to play again? Please press 1 for Yes and 0 for No.

- - - - - - - - - - - - - - -
All Output
```

Figure 12. Input and Output / Result for Test I.11

Result:

Fail: The program did not handle the erroneous data and a continuous loop asking user for new input was entered. This error was expected, and to improve the program in the future, measures should be put in place to handle erroneous data gracefully and to prompt user for new input only once.

Test I.12

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
&^

- - - - - - - - - - - - - - -
Sorry, I didn't get that. Please try again.



Do you want to play again? Please press 1 for Yes and 0 for No.

- - - - - - - - - - - - - - -
Sorry, I didn't get that. Please try again.



Do you want to play again? Please press 1 for Yes and 0 for No.

- - - - - - - - - - - - - - -
Sorry, I didn't get that. Please try again.



Do you want to play again? Please press 1 for Yes and 0 for No.

All Output
```

Figure 13. Input and Output / Result for Test I.12

Result:

Fail as above (test I.11): The program did not handle the erroneous data and a continuous loop asking user for new input was entered. This error was expected, and to improve the program in the future, measures should be put in place to handle erroneous data gracefully and to prompt user for new input only once.

Processing and Output Test Runs

Test PO.1

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
1

- - - - - - - - - - - - - - -


** shuffled deck:
* 4 * 2 * 3 * 5 * 4 * 1 * 3 * 6 * 1 * 6 * 5 * 2

To start, I will ask you to pick a card. The deck is numbered 1 to 12.

I will then ask you to pick another card.

If they match, the cards will be removed from the deck and represented using '99'.
 Otherwise, I wil ask you for another card.


Ready?

     ----       ----       ----       ----
  1.|    |   2.|    |   3.|    |   4.|    |
     ----       ----       ----       ----
     ----       ----       ----       ----
  5.|    |   6.|    |   7.|    |   8.|    |
     ----       ----       ----       ----
     ----       ----       ----       ----
  9.|    |  10.|    |  11.|    |  12.|    |
     ----       ----       ----       ----


** Current Card Deck:
*** 4 * 2 * 3 * 5 * 4 * 1 * 3 * 6 * 1 * 6 * 5 * 2

Please pick the card you want to turn over (1-12)
|
```

Print statements for testing purposes

Figure 14. Input and Output / Result for Test PO.1

Result:

Pass: The user opted to play the game by entering '1', and as expected, the play game function was called and the game info was output.

Test PO.2

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
0
Ok. Goodbye.
Program ended with exit code: 0
```

Figure 15. Input and Output / Result for Test PO.2

Result:

Pass: The user opted not to play the game by entering '0', and as expected, the goodbye message was printed and the program ended.

Test PO.3

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
1

- - - - - - - - - - - - - - -


** shuffled deck:
* 3 * 5 * 6 * 5 * 4 * 3 * 1 * 4 * 6 * 2 * 1 * 2
```

Figure 16. Input and Output / Result for Test PO.3 (Run 1)

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
1

- - - - - - - - - - - - - - -


** shuffled deck:
* 5 * 4 * 6 * 2 * 6 * 1 * 3 * 2 * 3 * 5 * 4 * 1
```

Figure 17. Input and Output / Result for Test PO.3 (Run 2)

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
1

- - - - - - - - - - - - - - -


** shuffled deck:
* 4 * 5 * 3 * 2 * 2 * 4 * 1 * 5 * 1 * 6 * 6 * 3
```

Figure 18. Input and Output / Result for Test PO.3 (Run 3)

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
1

- - - - - - - - - - - - - - -


** shuffled deck:
* 6 * 4 * 6 * 1 * 2 * 3 * 2 * 5 * 5 * 1 * 3 * 4
```

Figure 19. Input and Output / Result for Test PO.3 (Run 4)

```
- - - Hello, Welcome to the Card Matching Game - - -

Please enter your first name:
Christopher


Thanks Christopher. Do you want to start? Please press 1 for Yes and 0 for No.
1

- - - - - - - - - - - - - - -


** shuffled deck:
* 3 * 5 * 2 * 4 * 1 * 3 * 1 * 5 * 4 * 2 * 6 * 6
```

Figure 20. Input and Output / Result for Test PO.3 (Run 5)

Result:

Pass: The generate random card deck function produced five different card decks on separate runs of the program therefore proving that the function effectively shuffles the deck.

Test PO.4

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
** Current Card Deck:
*** 2 * 3 * 1 * 6 * 3 * 5 * 6 * 5 * 4 * 4 * 2 * 1

Please pick the card you want to turn over (1-12)
1

Please pick the next card you want to turn over (1-12)
2
    ----        ----        ----        ----
 1.| 02 |    2.| 03 |    3.| 00 |    4.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 9.| 00 |   10.| 00 |   11.| 00 |   12.| 00 |
    ----        ----        ----        ----
```

Figure 21. Input and Output / Result for Test PO.4 (Run 1)

```
** Current Card Deck:
*** 2 * 3 * 1 * 6 * 3 * 5 * 6 * 5 * 4 * 4 * 2 * 1

Please pick the card you want to turn over (1-12)
3

Please pick the next card you want to turn over (1-12)
4
    ----        ----        ----        ----
 1.| 00 |    2.| 00 |    3.| 01 |    4.| 06 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 9.| 00 |   10.| 00 |   11.| 00 |   12.| 00 |
    ----        ----        ----        ----
```

Figure 22. Input and Output / Result for Test PO.4 (Run 2)

```
** Current Card Deck:
*** 2 * 3 * 1 * 6 * 3 * 5 * 6 * 5 * 4 * 4 * 2 * 1

Please pick the card you want to turn over (1-12)
9

Please pick the next card you want to turn over (1-12)
10
     ----        ----        ----        ----
  1.| 00 |    2.| 00 |    3.| 00 |    4.| 00 |
     ----        ----        ----        ----
     ----        ----        ----        ----
  5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
     ----        ----        ----        ----
     ----        ----        ----        ----
  9.| 04 |   10.| 04 |   11.| 00 |   12.| 00 |
     ----        ----        ----        ----

!* Well done, you found a pair of 4's!! *!
```

Figure 23. Input and Output / Result for Test PO.4 (Run 3)

```
** Current Card Deck:
*** 2 * 3 * 1 * 6 * 3 * 5 * 6 * 5 * 99 * 99 * 2 * 1

Please pick the card you want to turn over (1-12)
5

Please pick the next card you want to turn over (1-12)
6
     ----        ----        ----        ----
  1.| 00 |    2.| 00 |    3.| 00 |    4.| 00 |
     ----        ----        ----        ----
     ----        ----        ----        ----
  5.| 03 |    6.| 05 |    7.| 00 |    8.| 00 |
     ----        ----        ----        ----
     ----        ----        ----        ----
  9.| 99 |   10.| 99 |   11.| 00 |   12.| 00 |
     ----        ----        ----        ----
```

Figure 24. Input and Output / Result for Test PO.4 (Run 4)

```
** Current Card Deck:
*** 1 * 3 * 5 * 3 * 6 * 4 * 5 * 1 * 2 * 6 * 4 * 2

Please pick the card you want to turn over (1-12)
11

Please pick the next card you want to turn over (1-12)
12
     ----       ----       ----       ----
  1.| 00 |   2.| 00 |   3.| 00 |   4.| 00 |
     ----       ----       ----       ----
     ----       ----       ----       ----
  5.| 00 |   6.| 00 |   7.| 00 |   8.| 00 |
     ----       ----       ----       ----
     ----       ----       ----       ----
  9.| 00 |  10.| 00 |  11.| 04 |  12.| 02 |
     ----       ----       ----       ----
```

Figure 25. Input and Output / Result for Test PO.4 (Run 5)

Result:

Pass: The graphical card deck displayed the correct values to the user based on the card deck array, for each of the five test runs, as highlighted using red boxes in the test screenshots.

Test PO.5

Screenshot Evidence from IDE: *testing print statements still shown with: *

```
** Current Card Deck:
*** 3 * 4 * 3 * 6 * 2 * 2 * 4 * 5 * 6 * 1 * 1 * 5

Please pick the card you want to turn over (1-12)
10

Please pick the next card you want to turn over (1-12)
11
     ----       ----       ----       ----
  1.| 00 |   2.| 00 |   3.| 00 |   4.| 00 |
     ----       ----       ----       ----
     ----       ----       ----       ----
  5.| 00 |   6.| 00 |   7.| 00 |   8.| 00 |
     ----       ----       ----       ----
     ----       ----       ----       ----
  9.| 00 |  10.| 01 |  11.| 01 |  12.| 00 |
     ----       ----       ----       ----

!* Well done, you found a pair of 1's!! *!


Your current score is: (1)
Your current attempt count is: (1)


** Current Card Deck:
*** 3 * 4 * 3 * 6 * 2 * 2 * 4 * 5 * 6 * 99 * 99 * 5

Please pick the card you want to turn over (1-12)
8

Please pick the next card you want to turn over (1-12)
12
     ----       ----       ----       ----
  1.| 00 |   2.| 00 |   3.| 00 |   4.| 00 |
     ----       ----       ----       ----
     ----       ----       ----       ----
  5.| 00 |   6.| 00 |   7.| 00 |   8.| 05 |
     ----       ----       ----       ----
     ----       ----       ----       ----
  9.| 00 |  10.| 99 |  11.| 99 |  12.| 05 |
     ----       ----       ----       ----

!* Well done, you found a pair of 5's!! *!


Your current score is: (2)
Your current attempt count is: (2)


** Current Card Deck:
*** 3 * 4 * 3 * 6 * 2 * 2 * 4 * 99 * 6 * 99 * 99 * 99
```

Figure 26. Input and Output / Result for Test PO.5 (Run 1)

```
    ----        ----        ----        ----
  1.| 00 |    2.| 00 |    3.| 02 |    4.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  9.| 00 |   10.| 00 |   11.| 00 |   12.| 02 |
    ----        ----        ----        ----

!* Well done, you found a pair of 2's!! *!


Your current score is: (1)
Your current attempt count is: (1)


** Current Card Deck:
*** 1 * 6 * 99 * 3 * 3 * 1 * 5 * 4 * 6 * 5 * 4 * 99

Please pick the card you want to turn over (1-12)
1

Please pick the next card you want to turn over (1-12)
6
    ----        ----        ----        ----
  1.| 01 |    2.| 00 |    3.| 99 |    4.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  5.| 00 |    6.| 01 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  9.| 00 |   10.| 00 |   11.| 00 |   12.| 99 |
    ----        ----        ----        ----
```

Figure 27. Input and Output / Result for Test PO.5 (Run 2)

```
    ----        ----        ----        ----
  1.| 99 |    2.| 00 |    3.| 99 |    4.| 03 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  5.| 03 |    6.| 99 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  9.| 00 |   10.| 00 |   11.| 00 |   12.| 99 |
    ----        ----        ----        ----

!* Well done, you found a pair of 3's!! *!


Your current score is: (3)
Your current attempt count is: (3)


** Current Card Deck:
*** 99 * 6 * 99 * 99 * 99 * 99 * 5 * 4 * 6 * 5 * 4 * 99

Please pick the card you want to turn over (1-12)
2

Please pick the next card you want to turn over (1-12)
9
    ----        ----        ----        ----
  1.| 99 |    2.| 06 |    3.| 99 |    4.| 99 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  5.| 99 |    6.| 99 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  9.| 06 |   10.| 00 |   11.| 00 |   12.| 99 |
    ----        ----        ----        ----
```

Figure 28. Input and Output / Result for Test PO.5 (Run 3)

```
    ----        ----        ----        ----
  1.| 00 |    2.| 00 |    3.| 04 |    4.| 04 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  9.| 00 |   10.| 00 |   11.| 00 |   12.| 00 |
    ----        ----        ----        ----

!* Well done, you found a pair of 4's!! *!


Your current score is: (1)
Your current attempt count is: (1)


** Current Card Deck:
*** 6 * 5 * 99 * 99 * 6 * 2 * 3 * 1 * 3 * 1 * 2 * 5

Please pick the card you want to turn over (1-12)
1

Please pick the next card you want to turn over (1-12)
2
    ----        ----        ----        ----
  1.| 06 |    2.| 05 |    3.| 99 |    4.| 99 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
  9.| 00 |   10.| 00 |   11.| 00 |   12.| 00 |
    ----        ----        ----        ----
```

Figure 29. Input and Output / Result for Test PO.5 (Run 4)

```
    ----        ----        ----        ----
 1.| 00 |    2.| 00 |    3.| 00 |    4.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 9.| 00 |   10.| 04 |   11.| 00 |   12.| 04 |
    ----        ----        ----        ----

!* Well done, you found a pair of 4's!! *!


Your current score is: (1)
Your current attempt count is: (1)


** Current Card Deck:
*** 3 * 1 * 6 * 5 * 2 * 5 * 2 * 3 * 1 * 99 * 6 * 99

Please pick the card you want to turn over (1-12)
1

Please pick the next card you want to turn over (1-12)
2
    ----        ----        ----        ----
 1.| 03 |    2.| 01 |    3.| 00 |    4.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 5.| 00 |    6.| 00 |    7.| 00 |    8.| 00 |
    ----        ----        ----        ----
    ----        ----        ----        ----
 9.| 00 |   10.| 99 |   11.| 00 |   12.| 99 |
    ----        ----        ----        ----
```

Figure 30. Input and Output / Result for Test PO.5 (Run 5)

Result:

Pass: The graphical card deck displayed the correct values to the user based on the card deck array, for each of the five test runs, as highlighted using red boxes in the test screenshots. The program changed the card deck values to 99 when the cards had been matched.

Test PO.6

Screenshot Evidence from IDE: *testing print statements still shown with: *

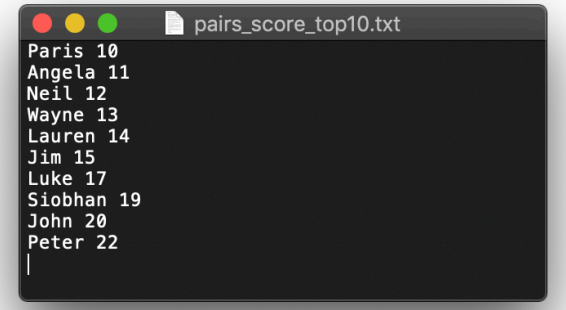


Figure 31. Leader board file before running the program for test PO.6

Figure 32. Leader board in program (showing current player Christopher's score of 6 as top score) for test PO.6

Result:

Pass: The program correctly read the pairs_score_top10.txt file and outputted them to the user correctly.

Test PO.7

Screenshot Evidence from IDE: *testing print statements still shown with: *

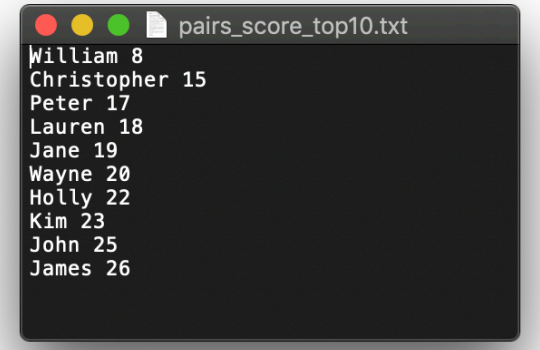


Figure 33. Leader board file before running the program for test PO.7

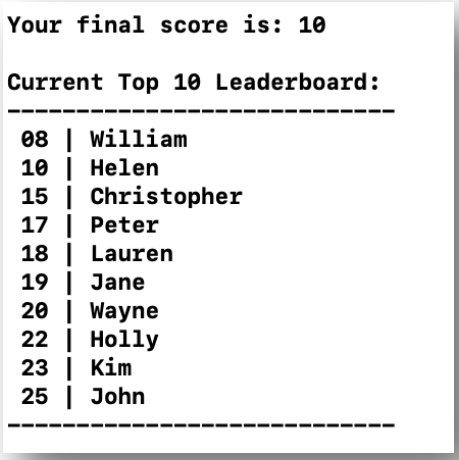


Figure 34. Leader board in program (showing current player Helen's score of 10 as second best score) for test PO.7

Result:

Pass: The program correctly read the pairs_score_top10.txt file, amended the current player's score of 10 to the arrays, and then correctly sorted the list in ascending order and outputted this to the user.

Test PO.8

Screenshot Evidence from IDE: *testing print statements still shown with: *

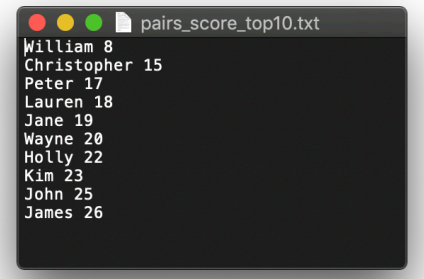


Figure 35. Leader board file before running the program for test PO.8

Figure 36. Leader board in program (showing current player Helen's score of 10 as second best score) for test PO.8

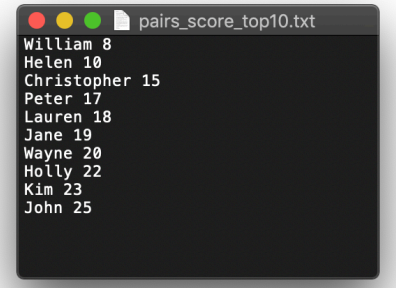


Figure 37. Leader board file showing the user's new score added correctly and only 10 records for test PO.8

Result:

Pass: The program correctly wrote the new score to the file and the file contains only the top-10 records as required.

1.4. Limitations of the Solution

The solution does not currently validate against the input of data of conflicting types, such as the input of a char (character) where a float or integer is expected. This currently causes the program to crash and enter into an infinite loop.

Future improvements to the program would be to implement input validation to check the data type of the input, and handle it accordingly, by catching the error and prompting the user to enter valid data.