



School of Computing and Mathematical Sciences

## CMSHN1201 Programming Workshop Referral Coursework 2000/2001

### Cards 'n' Dice game

Coursework : Exam weighting

Weighting towards all courseworks (%)

Learning Outcomes Being Assessed

Set by	Dr Paul Vickers (718) <i>p.vickers@livjm.ac.uk</i>
Moderated by	Mr Andrew Laws (603) <i>a.laws@livjm.ac.uk</i>
Hand-out date	
Hand-in date	
Expected hand-back date	
Expected feedback date(if different from hand-back date)	
Estimated number of hours of student effort	24 Hours

**Outline of Problem** To write a C program that will:

- Play a simple game of cards and dice with the user.

### Introduction

A program is required that will play two games with the user. The first is a simple game of cards in which the player and the dealer (the computer) are each dealt a hand of five cards. The hands are to be displayed on the screen together with their points value. An ace is worth one point and the Jack, Queen, and King are each worth ten points. Whoever has the most points wins the game.

The second game is a dice game. The rules are quite simple. The player rolls two dice. If he rolls 7 or 11 then he is immediately declared the winner. If he rolls 2, 3, or 12, then he is immediately declared the loser. If a roll of 4, 5, 6, 8, 9, or 10 is made then the value of the roll is called the **point**. The player then makes further rolls of the dice until either he makes the point (i.e., rolling the same value again) in which case he wins, or he rolls a 7 in which case he loses.

The player should choose which game to play from a simple menu:

```
Cards 'n' Dice
-----

1. Play cards
2. Play dice
3. Quit

Select: [ ]
```

**Menu option 1**

Choosing option 1 will cause the cards game to be played. First of all the pack must be randomised (shuffled) and then the player and the computer dealt 5 cards each. The game should look like this:

```

Cards 'n' Dice
-----

Player's hand      Dealer's Hand
-----
Nine of Diamonds   King of Clubs
Jack of Hearts     Two of Clubs
Three of Spades    Seven of Spades
Seven of Clubs     Seven of Hearts
Six of Diamonds    Five of Diamonds

Value : 35         Value : 31

Player won!

Hit any key to return to menu

```

**Menu option 2**

Choosing this option starts the dice game, and the screen should look like this:

```

Cards 'n' Dice
-----

Point to make      [__]

Player rolled      [__]

Hit a key to make first roll

```

Upon hitting a key, the first roll is made. If the player rolls a 7 or 11, then the screen will change:

```

Cards 'n' Dice
-----

Point to make      [__]

Player rolled 7    [2,5]

Player wins!!!

Hit any key to return to menu

```

If instead the player rolls a 2, 3, or 12, the he would lose on the first roll and the screen would look like this (the numbers in the square brackets show the value of each die):

```

Cards 'n' Dice
-----

Point to make      [__]

Player rolled 2    [1,1]

Player loses

Hit any key to return to menu

```

If a **point** score is rolled then the screen would show this:

```

Cards 'n' Dice
-----

Point to make      [ 6]

Player rolled 6   [4,2]

Hit a key to roll again

```

The first roll was a 6 and so that becomes the point the player has to make. The player continues to make rolls of the dice until either the point is rolled again and he wins, or a 7 is rolled and he loses; in either case a suitable message should be displayed.

### Menu option 3

When the player chooses to quit, then a final message is displayed showing his overall score:

```

Cards 'n' Dice
-----

You won 1/1 card games and 4/5 dice games

Hit any key to end

```

### Your task

Your task is to complete the program whose outline is given in Appendix A. You will notice that the program uses seven new functions:

- `void openPack (card [52]) ;`
- `void shuffle (card [52]) ;`
- `void deal (card [52], card [5], card [5]) ;`
- `void displaycard (card) ;`
- `int rollDice (int*, int*) ;`
- `void clearClient (int, int, int) ;`
- `int handValue (card [5]) ;`

The functions make various use of the **struct card**. A card is simply a record that has three fields, one for the name of a card, one for its suit, and one for its point value. The definition for a card is given in the header file `cards.h` (see below for where to find this file). You must code all the functions except for `openPack` and `clearClient` which we have written for you. All the functions reside in `functions.h`.

#### **openPack**

This function initialises a new pack of cards (array of card structs) by making the first card equal to "Ace" of "Hearts", the second "Two" of "Hearts", and so on up the "King" of "Spades". We have written this function for you. The card and suit names are stored in two string arrays (see `cards.h`) called `faces` and `suits`.

#### **shuffle**

This function shuffles a pack of cards.

**deal**

This function deals five cards to the player and the computer. The cards are stored in the two arrays `playersHand` and `dealersHand` (see `cards.h`).

**displayCard**

Simply displays the value of a card on the screen.

**rollDice**

This uses the random number generator (see below) to randomly assign a value between 1 and seven to each of its two reference arguments `die1` and `die2`. The values of the two dice added together are returned by the function.

**clearClient**

This function clears an area of screen between `x`, `y1` and `x`, `y2`. We have written this for you.

**handValue**

Works out how much a hand of cards is worth and returns that value.

**Guidance****Random numbers**

The program uses C's random number generator function `rand ()` to control the card shuffling process and the dice throws. `rand` returns a random number from 0 up to the largest integer. If you want a random number between, say, 1 and 10, then you would write:

```
number = 1 + rand () % 10 ;
```

Before you use `rand`, it must itself be randomised, otherwise each time the program is run, it will always generate the same sequence of random numbers. We do this with the following statement:

```
srand (time (NULL)) ;
```

Don't worry about how that works – just use it.

**Output**

To make the program more pleasant to use, we shall be using the C screen layout functions `clrscr`, `clrcol`, and `gotoxy`. Examples of their use are provided in the bits of the program I have already coded.

**Online resources**

All the files associated with the coursework can be downloaded from the module web site at:

<http://www.cms.livjm.ac.uk/paulvickers/HN1201/resource.htm>

There you will also find a fully working version of the program, called `cards.exe`. Running the fully-working version will give you a feel for what the specification really means. Make sure you copy the program to your own folder before attempting to execute it. The files available then are:

- `Cards.cpp` (the program outline given in Appendix A)
- `Cards.h` (a header file containing the card struct definitions)
- `Functions.h` (a header file in which you will put the various functions)
- `Cards.exe` (a working version of the program)
- `Referral2000_01.pdf` (a copy of this document).

**Getting help**

If you have any problems, then you may email [p.vickers@livjm.ac.uk](mailto:p.vickers@livjm.ac.uk), though I cannot promise to give exhaustive help.

## General Instructions

### Resources Required

- Access to C compiler
- Access to [www.cms.livjm.ac.uk/paulvickers](http://www.cms.livjm.ac.uk/paulvickers)

### Assessment

Marks will be awarded for:

Assessment Criteria	Weight
<ul style="list-style-type: none"> <li>• Compilation without errors</li> <li>• Producing the correct results with output exactly as specified.</li> <li>• Structure of code matching outline design.</li> <li>• Use of meaningful identifiers and layout of code.</li> <li>• Use of correct data types</li> <li>• Good test plan and testing</li> </ul>	<b>100%</b>

To get a pass grade, I will be looking for:

- A program that meets all the stated requirements
- Good use of interactive dialogue and screen layout to make the program user-friendly (see the sample program on the web site for an example of this)
- Clear and well-structured outline designs (where required)
- Clear and concise coding that matches the outline designs and which conforms to the standards used in the course text book
- Effective naming and labelling of identifiers (where new ones are introduced)
- Evidence of a thorough test plan with documentation of expected and actual results.

A **fail grade** will be awarded when a significant majority of the criteria are not met through major flaws and/or omission.

You should hand in a diskette with your C program on it. I do not need copies of the input or output files.

## Appendix A

### Program outline

```

/* CMSHN1201 Programming Workshop - Referral Coursework 2000/01          */
/* Playing card shuffler and dealer simulator                            */
/* Description.....                                                    */
Seed Rand function
Initialise a new pack of cards
do
{
  Clear the screen
  Display game heading
  Display menu
  Prompt for and accept user's choice
  switch (users choice)
  {
    1 : ---Play cards
      Shuffle the pack
      Deal 5 cards to player and dealer
      Clear the playing area on screen
      Display hand contents heading
      Display player's hand of cards and value
      Display dealer's hand of cards and value
      if player's hand worth more than dealer's
      {
        Display 'player won' message
        Add 1 to number of card games won
      }
      else if players hand worth less than dealer's
        Display 'Dealer won' message
      else
        Display 'draw' message
      Prompt for and accept any key to return to menu
      Add 1 to number of card games played
    2 : ---Play dice
      Clear playing area on screen
      Display headings for point to make and dice value thrown
      Prompt for and accept any key to make first roll of dice
      roll the dice and store sum of the roll
      display dice roll and sum
      switch (sum)
      {
        7, 11 : Set gamestatus = 1 (win on first roll)
        2, 3, 12 : Set gamestatus = 2 (lost on first roll)
        default : Set gamestatus = 0
                  make myPoint = sum
                  display point to make (myPoint)
      }
      while (not won or lost) (gamestatus is 0)
      {
        Prompt for and accept any key to roll again
        roll the dice and store sum
        Display dice roll and sum
        if sum = point to make
          set gamestatus = 1 (a win)
        else if sum is 7
          set gamestatus = 2 (a seven loses)
      }
      if player won
      {
        Add 1 to number of dice games won
        Display 'Player wins' message
      }
      else

```

```

        Display 'Player loses' message
        prompt for and accept any key to return to menu
        Add 1 to number of dice games played
    }
}
while (user not choose Quit)
    Display 'You won x/y card games and n/m dice games' message
    Prompt for and accept any key to quit program */
#include <stdio.h>
#include <stdlib.h>      /* Holds definition of 'rand' function      */
#include <time.h>        /* Get time of day to seed 'rand' function      */
#include <conio.h>
#include <string.h>
#include "cards.h"
#include "functions.h"

/* Define the starting coordinates for screen output      */
#define X 25
#define Y 9

/* Function prototypes */
void openPack (card [52]) ;
void shuffle (card [52]) ;
void deal (card [52], card [5], card [5]) ;
void displaycard (card) ;
int rollDice (int*, int*) ;
void clearClient (int, int, int) ;
int handValue (card [5]) ;

main ()
{
    card pack [52] ;
    card playersHand [5] ;
    card dealersHand [5] ;

    int userResponse;
    int counter ;
    int die1,
        die2,
        sum,
        myPoint,
        gameStatus,
        x,
        Y,
        cardsWon = 0,
        cardsPlayed = 0,
        diceWon = 0,
        dicePlayed = 0 ;

    /* Seed the rand function to ensure 'true' random numbers */
    srand (time (NULL)) ;
}

```

## CMSHN1201 Programming Workshop Referral Coursework Cover Sheet

Complete and attach this form to your coursework submission.

**THIS IS NOT A RECEIPT.**

Name	Registration number

This work is for the attention of

**Dr Paul Vickers, CMS**

### Checklist:

The grade I realistically think this piece of work is worth is:

### I have included the following items (tick boxes when complete):

Virus-free diskette containing C source code:

Listing on collated stationery of your source program

Test plan and expected and actual results:

Signed

Date