ASSIGNMENT 9

GENE-121-2017-3F-ASSN-09-Description

Fall 2017

CONTENTS

INTRODUCTION

This assignment is to be done individually.

The problems in this assignment are an exercise to:

- solve a problem using a recursive function
- create and use a class including use of the implicit parameter and an array of objects

QUESTION 1 – CARDHAND CLASS

Ryan is practicing for the World Series of Poker. He needs your assistance building a card game simulator. Ryan will be creating the code to handle dealing the cards. However, he needs a data structure to hold the cards after they have been dealt.

Build a CardHand class that stores a series of cards, to represent the hand of cards that a player holds. The class needs to handle the ability to play different games, for which a different number of cards is required to be held in the hand.

A template file has been provided to you with an enumerated data type Suit, a class for storing individual cards, and Ryan's card-dealing function.

TEMPLATE FILE

Use the given Template.cpp file for writing your code. You will need to add member variables and functions to the CardHand class.

Note that the class function definitions are placed below the main function. The shell of the GetHandSize and AddCard functions have been included to give you an example to follow. You will need to modify the code to make these functions work as specified.

DESIGN

Consider and submit answers to the following questions:

- What are the most cards that the hand could ever hold? Explain your reasoning.
- What information does the class need to store in order to handle the fact that the class can be used to play different games, which require the player to hold different number of cards for each game?

CARD CLASS

We have provided you with a Card class in the template file. This data type is to be used to store individual cards. Each card consists of two parts: its value (from 1 to 13) and its suit (CLUBS, DIAMONDS, HEARTS, SPADES).

CARDS MEMBER VARIABLE

The class should store the cards as a private member array of Cards.

DEFAULT CONSTRUCTOR

The default constructor should initialize the hand such that it contains zero cards and has a hand size of 5. This means that it is ready to play a game where the player gets 5 cards. The hand size is not permitted to change after the object is initialized.

DATA CONSTRUCTOR

The data constructor allows the hand to be setup to play with a different number of cards in the hand. If the requested number of cards for the hand is invalid, use a hand size of 5. The hand size is not permitted to change after the object is initialized.

The hand should still contain zero cards if the data constructor is used.

PRIVATE MEMBER FUNCTION – SORTCARDS

Write a private function that sorts the cards in descending order. Sort first by value and then by suit. Note that the suits ranked from highest to lowest are: SPADES, DIAMONDS, HEARTS, CLUBS. Ace is considered the lowest value. The following shows a correct sorting:

```
J  HEARTS
7  SPADES
5  SPADES
5  CLUBS
A  SPADES
```

ACCESSOR – GETHANDSIZE METHOD

This accessor allows the caller to retrieve the number of cards that the hand can hold.

MUTATOR – ADDCARD METHOD

The AddCard method accepts a Card and adds it to the hand. Only add the card if there is room for the card in the hand.

BETTERHAND METHOD

Create a public method (function) within the class that receives another CardHand object. It will compare the current CardHand (this) with the passed CardHand and return a value indicating which hand is better. If this CardHand is better, it returns +1. If the passed CardHand is better, it returns -1. If they are equivalent, it returns 0.

It is up to you how the CardHands are to be compared. Include an explanation of your ranking method as a comment in your code. An example of a method would be to add up the values of all the cards in the hand and the hand with the higher value is the better hand. Another method would be to count the number of face cards in the hand and rank the hand higher that has the greater number of face cards.

ACCESSOR – PRINTHAND METHOD

Receives an ostream parameter (passed by reference). This means that the user could pass in cout or fout to this method.

Outputs all the cards in the hand. The cards are to be printed in a sorted descending order. There should a line demarcating the start of the hand and a line for the end of the hand.

The following is an example of the expected output from this method.

```
start of hand
J HEARTS
7 SPADES
5 SPADES
5 CLUBS
A SPADES
end of hand
```

MAIN FUNCTION

Write a main function that uses the given DealCards function to deal out hands to four players. Then print out the hands with the cards in each hand in sorted order.

Output which of the four players has the best hand. You are not required to consider ties. In other words, if multiple players have the best hand, you are only required to output one of those players.

Note that you should not require and should not use the assignment operator (=) on objects of the CardHand class. This is because we have not covered how to use the assignment operator on classes yet. By default, the assignment operator does not work in the expected fashion. We will be covering this topic later.

The output can be either to the console or to a file. Submit your code with output.

QUESTION 2 – RECURSION (C++)

On Assignment 2 you programmed a Guess a Number game. In this problem, you will repeat the exercise. This time, you will extend the game using a recursive function.

The program will first accept two numbers indicating the range of possible numbers that the user is thinking of. Then, the program will continuously guess a number until either it has correctly guessed the number or it has determined that the user is not telling the truth about their secret number. If the program correctly guesses the number, it is to also output the number of guesses that it took.

You can assume that the secret number must be able to be stored in an int variable. You must use a recursive function to guess the secret number.

Submit your code along with output for your program for the following two test cases:

1. The program takes 5 or more guesses to correctly guess the number.
2. The program is unable to guess the number because the user is not telling the truth about their secret number.

SAMPLE OUTPUT

```
Recursive Guess-A-Number
Think of a secret number, which this program will attempt to guess.
Enter the lowest possible number: 1
Enter the highest possible number: 16

Is the number 8?
Enter (Y)es, (L)ower, or (H)igher: H

Is the number 12?
Enter (Y)es, (L)ower, or (H)igher: L

Is the number 10?
Enter (Y)es, (L)ower, or (H)igher: Y

Yaay, I correctly guessed your secret number in 3 guesses!
```