```
// Specification file for the Dealer class
#ifndef DEALER_H
#define DEALER_H
#include <string>
#include "Die.h"
using namespace std;

class Dealer
{
private:
   Die die1;             // Object for die #1
   Die die2;             // Object for die #2
   int die1Value;        // Value of die #1
   int die2Value;        // Value of die #2

public:
   Dealer();             // Constructor
   void rollDice();      // To roll the dice
   string getChoOrHan(); // To get the result (Cho or Han)
   int getDie1Value();   // To get the value of die #1
   int getDie2Value();   // To get the value of die #2
};
#endif
```

```
// Specification file for the Die class
#ifndef DIE_H
#define DIE_H

class Die
{
private:
   int sides;   // Number of sides
   int value;   // The die's value

public:
   Die();
   Die(int);        // Constructor
   void roll();     // Rolls the die
   int getSides();  // Returns the number of sides
   int getValue();  // REturns the die's value
};
#endif
```

```
// Specification file for the Player class
#ifndef PLAYER_H
#define PLAYER_H
#include <string>
using namespace std;

class Player
{
private:
   string name;         // The player's name
   string guess;        // The player's guess
   int points;          // The player's points

public:
   Player(string);      // Constructor
   void makeGuess();    // Causes player to make a guess
   void addPoints(int); // Adds points to the player
   string getName();    // Returns the player's name
   string getGuess();   // Returns the player's guess
   int getPoints();     // Returns the player's points
};
#endif
```

```
// Implementation file for the Dealer class
#include "Dealer.h"
#include "Die.h"
#include <string>
using namespace std;

//*****************************************
// Constructor                            *
//*****************************************
Dealer::Dealer()
{
   // Set the intial dice values to 0.
   // (We will not use these values.)
   die1Value = 0;
   die2Value = 0;
}

//*****************************************
// The rollDice member function rolls the  *
// dice and saves their values.            *
//*****************************************
void Dealer::rollDice()
{
   // Roll the dice.
   die1.roll();
   die2.roll();

   // Save the dice values.
   die1Value = die1.getValue();
   die2Value = die2.getValue();
}

//*****************************************
// The getChoOrHan member function returns *
// the result of the dice roll, Cho (even) *
// or Han (odd).                           *
//*****************************************
string Dealer::getChoOrHan()
{
   string result; // To hold the result

   // Get the sum of the dice.
   int sum = die1Value + die2Value;

   // Determine even or odd.
   if (sum % 2 == 0)
      result = "Cho (even)";
   else
      result = "Han (odd)";

   // Return the result.
   return result;
}

//*****************************************
// The getDie1Value member function returns *
// the value of die #1.                    *
//*****************************************
int Dealer::getDie1Value()
{
   return die1Value;
}

//*****************************************
// The getDie2Value member function returns *
// the value of die #2.                    *
//*****************************************
int Dealer::getDie2Value()
{
   return die2Value;
```

```
}
```

```
// Implememtation file for the Die class
#include <cstdlib>     // For rand and srand
#include <ctime>       // For the time function
#include "Die.h"
using namespace std;

Die::Die()
{
   sides = 6;
   roll();
}

//*****************************************************
// The constructor accepts an argument for the number   *
// of sides for the die, and performs a roll.           *
//*****************************************************
Die::Die(int numSides)
{
   // Get the system time.
   unsigned seed = time(0);

   // Seed the random number generator.
   srand(seed);

   // Set the number of sides.
   sides = numSides;

   // Perform an initial roll.
   roll();
}

//*****************************************************
// The roll member function simulates the rolling of    *
// the die.                                             *
//*****************************************************
void Die::roll()
{
   // Constant for the minimum die value
   const int MIN_VALUE = 1;   // Minimum die value

   // Get a random value for the die.
   value = (rand() % (sides - MIN_VALUE + 1)) + MIN_VALUE;
}

//*****************************************************
// The getSides member function returns the number of   *
// for this die.                                        *
//*****************************************************
int Die::getSides()
{
   return sides;
}

//*****************************************************
// The getValue member function returns the die's value.*
//*****************************************************
int Die::getValue()
{
   return value;
}
```

```
// Implementation file for the Player class
#include "Player.h"
#include <cstdlib>
#include <ctime>
#include <string>
using namespace std;

//*********************************************
// Constructor                                *
//*********************************************
Player::Player(string playerName)
{
   // Seed the random number generator.
   srand(time(0));

   name = playerName;
   guess = "";
   points = 0;
}

//*********************************************
// The makeGuess member function causes the    *
// player to make a guess, either "Cho (even)" *
// or "Han (odd)".                            *
//*********************************************
void Player::makeGuess()
{
   const int MIN_VALUE = 0;
   const int MAX_VALUE = 1;

   int guessNumber;  // For the user's guess

   // Get a random number, either 0 or 1.
   guessNumber = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;

   // Convert the random number to Cho or Han.
   if (guessNumber == 0)
      guess = "Cho (even)";
   else
      guess = "Han (odd)";
}

//*********************************************
// The addPoints member function adds a        *
// specified number of points to the player's  *
// current balance.                           *
//*********************************************
void Player::addPoints(int newPoints)
{
   points += newPoints;
}

//*********************************************
// The getName member function returns a       *
// player's name.                             *
//*********************************************
string Player::getName()
{
   return name;
}

//*********************************************
// The getGuess member function returns a      *
// player's guess.                            *
//*********************************************
string Player::getGuess()
{
   return guess;
}
```

```
//*******************************************
// The getPoints member function returns a     *
// player's points.                            *
//*******************************************
int Player::getPoints()
{
   return points;
}
```

```
// This program simulates the game of Cho-Han.
#include <iostream>
#include <string>
#include "Dealer.h"
#include "Player.h"
using namespace std;

// Function prototypes
void roundResults(Dealer &, Player &, Player &);
void checkGuess(Player &, Dealer &);
void displayGrandWinner(Player, Player);

int main()
{
   const int MAX_ROUNDS = 5;  // Number of rounds
   string player1Name;        // First player's name
   string player2Name;        // Second player's name

   // Get the player's names.
   cout << "Enter the first player's name: ";
   cin >> player1Name;
   cout << "Enter the second player's name: ";
   cin >> player2Name;

   // Create the dealer.
   Dealer dealer;

   // Create the two players.
   Player player1(player1Name);
   Player player2(player2Name);

   // Play the rounds.
   for (int round = 0; round < MAX_ROUNDS; round++)
   {
      cout << "---------------------------\n";
      cout << "Now playing round " << (round + 1)
          << endl;

      // Roll the dice.
      dealer.rollDice();

      // The players make their guesses.
      player1.makeGuess();
      player2.makeGuess();

      // Determine the winner of this round.
      roundResults(dealer, player1, player2);
   }

   // Display the grand winner.
   displayGrandWinner(player1, player2);
   return 0;
}

//************************************************
// The roundResults function determines the results *
// of the current round.                           *
//************************************************
void roundResults(Dealer &dealer, Player &player1, Player &player2)
{
   // Show the dice values.
   cout << "The dealer rolled " << dealer.getDie1Value()
        << " and " << dealer.getDie2Value() << endl;

   // Show the result (Cho or Han).
   cout << "Result: " << dealer.getChoOrHan() << endl;

   // Check each player's guess and award points.
   checkGuess(player1, dealer);
   checkGuess(player2, dealer);
```

```
}

//************************************************
// The checkGuess function checks a player's guess  *
// against the dealer's result.                     *
//************************************************
void checkGuess(Player &player, Dealer &dealer)
{
   const int POINTS_TO_ADD = 1;  // Points to award winner

   // Get the player's guess
   string guess = player.getGuess();

   // Get the result (Cho or Han).
   string choHanResult = dealer.getChoOrHan();

   // Display the player's guess.
   cout << "The player " << player.getName()
        << " guessed " << player.getGuess() << endl;

   // Award points if the player guessed correctly.
   if (guess == choHanResult)
   {
      player.addPoints(POINTS_TO_ADD);
      cout << "Awarding " << POINTS_TO_ADD
           << " point(s) to " << player.getName()
           << endl;
   }
}

//************************************************
// The displayGrandWinner function displays the     *
// game's grand winner.                             *
//************************************************
void displayGrandWinner(Player player1, Player player2)
{
   cout << "----------------------------\n";
   cout << "Game over. Here are the results:\n";

   // Display player #1's results.
   cout << player1.getName() << ": "
       << player1.getPoints() << " points\n";

   // Display player #2's results.
   cout << player2.getName() << ": "
       << player2.getPoints() << " points\n";

   // Determine the grand winner.
   if (player1.getPoints() > player2.getPoints())
   {
      cout << player1.getName()
           << " is the grand winner!\n";
   }
   else if (player2.getPoints() > player1.getPoints())
   {
      cout << player2.getName()
           << " is the grand winner!\n";
   }
   else
   {
      cout << "Both players are tied!\n";
   }
}
```