**Data Structures (2028C) Fall 2017 – Lab 3**

Topics covered: Arrays, Pointers, OOP, File I/O

Due date: 11th September 2017

**Objective:**

The objective for this lab is to develop a generic program for playing dice games using the concepts from Lab 2. The number of dice, number of players and the game logic should be customizable. The scores from each game should be persisted into a text file.

**Starter Code:**

DiceGame.cpp contains the starter code for this lab. Understand the code and class hierarchies in this file before you move onto the programming tasks. Also, remember to test your program at the end of each programming task.

**Task 1:**

The following program lets you generate pseudo-random numbers. Each time you run the program, you will see a different number on the output console.

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

int main(){

srand((unsigned)time(0));

cout<<rand()<<endl;

}

Utilize this example to complete the definition of “Dice” class. The “Dice” class represents an n-sided die. The roll of a die can result in any number between 1 and n. The number of sides should be obtained from the user at run-time. Test your code with different values of n before you move onto the next programming task.

**Task 2:**

The “Player” class represents a game player and contains two private variables – name and score.

Add a parameterized constructor to initialize these variables and complete the definition of accessor methods – getName() and getScore().

Define another class method to update the player’s score as the game proceeds.

**Task 3:**

The abstract class “DiceGame” will serve as the base class for implementing any kind of dice game. This class should provide support of managing the type of dice, the players of the game and recording the scores of players.

In this task, you will define two functions inside the “DiceGame” class as follows:

* initPlayers(): Function to accept the name and score of each player from the user. Creates instances of the “Player” class and adds them to an array that is referenced by the pointer “players”.
* displayScores(): Function to display the name and score of each player in the format “<player\_name>:<player\_score>”.

Figure out a way to test these methods before moving onto the next programming task. “DiceGame” is an abstract class and you cannot instantiate this class directly.

**Task 4:**

In this programming task, you will inherit from the abstract class “DiceGame” and implement a simple dice game called Knock Out. The rules of the game are as follows:

* At the beginning, N dice are rolled and the numbers on the dice are summed up. The resulting score is known as the Knock Out score.
* Following this, the players take turns to roll N dice. If the numbers on the dice add up to the Knock Out score, then the player gets knocked out.
* The remaining players continue to roll the dice and the game proceeds until only one player remains – the winner.

Accept the following information from the user:

* number of players
* number and type of dice (four 6-sided dice, three 4-sided dice, etc.)

Utilize a parameterized constructor to initialize the member variables of “DiceGame”. The logic of the game should be implemented inside the play() method. You can choose to reward the winner with as many points as you like. Update the player’s score in the corresponding instance of the Player class.

The invocation of all class methods from main() should happen through the base class pointer “game”.

**Task 5:**

In this programming task, you will inherit from the abstract class “DiceGame” and implement another simple dice game called the Boston dice game. In this game, each player rolls N dice. Each time, the dice with the highest number is preserved and the remaining dice are rolled. In the end, the player with the highest score wins.

Assume that the game is played with three 6-sided dice. The first roll results in 3, 6 and 4. The die with 6 is preserved and the remaining dice are rolled again. The second roll results in 5 and 2. The die with 5 is preserved and the other die is rolled again. The last roll result in 3 and the final score of the player is 6 + 5 + 3 = 14.

Accept the following information from the user:

* number of players
* number and type of dice (four 6-sided dice, three 4-sided dice, etc.)
* type of dice game (knock-out or Boston dice game)

The last input parameter should decide whether the class implementing Knock Out should be instantiated or the class implementing the Boston Dice Game should be instantiated. Either way, the base class pointer “game” should be utilized for all method invocations from the main() method.

**Task 6:**

Now that you have two implementations of the dice game, create a file based score card to capture the name and score of all players involved in the games.

The following program lets you write into a text file.

#include <iostream>

#include <fstream>

using namespace std;

int main () {

ofstream myfile ("example.txt");

if (myfile.is\_open())

{

myfile << "This is a line.\n";

myfile << "This is another line.\n";

myfile.close();

}

else cout << "Unable to open file";

return 0;

}

Add a method to the “DiceGame” class called writeScoresToFile() that utilizes the above logic to write the name and scores of the players into a text file at the end of a game. Use the format “<player\_name>:<player\_score>” for each line.

Make sure that each call to writeScoresToFile() appends content to the text file rather than overwriting its contents.

**Task 7:**

The following program lets you read from a text file.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main () {

string line;

ifstream myfile ("example.txt");

if (myfile.is\_open())

{

while ( getline (myfile,line) )

{

cout << line << '\n';

}

myfile.close();

}

else cout << "Unable to open file";

return 0;

}

Add a method getHighestScore() to the “DiceGame” class. This method should read from the scorecard that you created in the previous task and return the name and score of the player with the highest score.

The atoi() method from stdlib.h lets you convert a character sequence to an integer You might need this method to complete the task.

**Lab Submission:**

Turn in the completed DiceGame.cpp and the score card (.txt file) generated by your program.

**Lab Grading:**

Task 1: 5 points

Task 2: 5 points

Task 3: 10 points

Task 4: 20 points

Task 5: 20 points

Task 6: 15 points

Task 7: 15 points

Use of adequate comments and indentation: 10 points

No credits will be awarded for this lab if the program does not compile.

**Bonus Points:** Additional 20 points will be awarded to students who complete all programming tasks in the lab.