1. ****What is an implicit parameter? How does it differ from an explicit parameter?

The object on which a member function is applied is the implicit parameter. Every member function has an implicit parameter. This can be seen in a line of code such as “implicit\_parameter.length”. Explicit parameters of a member function are listed in the function deﬁnition. It can be seen in the line of code “function(explicit\_parameter)”.

1. Using the **[English\_length class](https://www.cs.drexel.edu/~mcs172/Sp14/notes/10.3_class_intro/files/English_length.h" \t "wenk)**[given in the Course Notes](https://www.cs.drexel.edu/~mcs172/Sp14/notes/10.3_class_intro/files/English_length.h" \t "wenk), write a method **isGreaterThan(const English\_length & L);** that returns **true** if the implicit parameter is a longer length than the explicit parameter, and **false** otherwise. You should turn in the modified class header and the implementation of the method found in**English\_length.cpp** (you do not have to turn in the entire implementation file).

bool English\_length::isGreaterThan(English\_length & L) const

{

bool test = false;

if (L.totalInches() < this->totalInches())

{

test = true;

}

return test;

}

1. Using the method you just implemented, write a subprogram called **min()** that takes two **English\_length** parameters and returns whichever is smallest.

#include "English\_length.h"

#include <iostream>

using namespace std;

English\_length min(English\_length & L, English\_length & R);

int main(void)

{

English\_length L;

English\_length R;

English\_length answer;

L.add\_inches(5);

R.add\_inches(10);

bool test;

answer = min(L, R);

cout << answer.toString();

int go;

cin >> go;

return 0;

}

English\_length min(English\_length & L, English\_length & R)

{

if (L.isGreaterThan(R))

{

return L;

}

return R;

}

1. Use **enum** to create a new data type called **ChineseZodiacSign** that can only take the values of the signs of the Chinese zodiac (which are, in order: Rat, Ox, Tiger, Rabbit, Dragon, Snake, Horse, Sheep, Monkey, Rooster, Dog, Pig)

enum Zodiacs { Rat, Ox, Tiger, Rabbit, Dragon, Snake, Horse, Sheep, Monkey, Rooster, Dog, Pig };

1. Using the type created above, write a subprogram with prototype

**ChineseZodiacSign nextSign(ChineseZodiacSign cz);**

that returns the next astrological sign of the Chinese Zodiac. Watch for rollovers.

ChineseZodiacSign nextSign(ChineseZodiacSign cz)

{

switch (cz)

{

//Ox, Tiger, Rabbit, Dragon, Snake, Horse, Sheep, Monkey, Rooster, Dog, Pig

case Rat:

return Pig;

break;

case Tiger:

return Rat;

break;

case Rabbit:

return Tiger;

break;

case Dragon:

return Rabbit;

break;

case Snake:

return Dragon;

break;

case Horse:

return Snake;

break;

case Sheep:

return Horse;

break;

case Monkey:

return Sheep;

break;

case Rooster:

return Monkey;

break;

case Dog:

return Rooster;

break;

case Pig:

return Dog;

break;

default:

cout << "It doesn't work.";

break;

}

}

**Programming Assignment**

**Background**

This is the first part of a sequence of assignments that deal with creation of an online movie information service, which allows people to look up information about movies and the actors that appear in them. For this part, you will focus on the movies.

**A Class**

For this assignment, you are to make use of a partially completed class, [**Movie**](https://www.cs.drexel.edu/~mcs172/Sp14/assignments/HW1/Movie.h)**(https://www.cs.drexel.edu/~mcs172/Sp14/assignments/HW1/Movie.h)**.

For our purposes, every movie has a *title* (e.g. "Rat Race"), a *director* (e.g. "Alfred Hitchcock"), a *year of release* (e.g., 1989), and a *rating* established by the [Motion Picture Association of America](http://www.mpaa.org/) (http://www.mpaa.org/) : (either **G** (General Audiences; all ages admitted), **PG** (Parental Guidance suggested; some material may not be suitable for children), **PG-13** (Parents Strongly Cautioned; some material may be inappropriate for children under 13), **R** (Restricted; under 17 requires accompanying parent or adult guardian), **NC-17** (Never Caught dead in this one), or **NR** (Not Rated)). Notice that we have created a new type called **Movie\_Rating** using **enum** to designate the MPAA's movie rating.

The first part of your assignment is to complete the implementation of the **Movie** class by implementing all of the methods. The default constructor should initialize a movie object to the "Null" movie (no title, etc.). The implementation for the class should be stored in a separate file with the **.cpp** extension. The header should be kept in a file with the **.h**extension. You will need to put other things in both files, such as **#include** statements and directives to prevent multiple compilation.

**The Program**

You are to write a program that tests the **Movie** class by performing the following tasks:

1. Write a subprogram (not a method) **void promptForMovie(Movie & myMovie);** that prompts the user for movie information, and returns a **Movie** object to the calling program. When this command is executed, the user will be asked to supply the attributes for the movie.
2. Write a method **void output(ostream & out);**that will display movie information as follows:

|  |
| --- |
| **Movie: Harold and Maude Director: Hal Ashby Year: 1971 Rating: R IMDB URL: http://www.imdb.com/title/tt0067185/** |

1. Have your program create 5 movies. One should be created using the "title only" constructor, and updating info with mutators. The remainder should use the**promptForMovie()** routine. All 5 movies used for testing your program must be different (and *real* movies - you should either obtain or verify your information from the[Internet Movie Database](http://www.imdb.com/) (http://www.imdb.com/) ). Make sure you use movies with a variety of ratings. (NOTE: you may not be able to find MPAA rating information at the IMDB site, but you can find it at the [MPAA Movie Ratings page](http://www.mpaa.org/film-ratings/) (http://www.mpaa.org/film-ratings/) .)
2. Print your list of movies on the screen and to a file. Submit this file with your assignment.

Your program (and all subsequent programs) must use good style, including proper variable names, use of indentation and whitespace, and commenting. Students are advised to follow the style guidelines supplied by the Horstmann text in Appendix A. (You should also consult the on-line version found on the course web page which contains instructors' editorials).

Note that for this (and all) programs, you must supply appropriate documentation.