Overview of Suggested Solutions

- Two full-credit versions are provided
  - full_credit: As specified using enums
  - full_credit_mvc: MVC pattern using classes
- Bonus / extreme_bonus (and suggested solutions to sprint 2 & 3) are based on full_credit_mvc
- Every directory includes:
  - A ./test_all script that builds (via make) and runs every regression test
  - ‘make rebuild’ rule to recompile everything
  - ‘make debug’ rule to recompile all for the debugger
  - ‘make div’ rule to create a visual separator

Homework #4 Sample Solutions

Full Credit and Bonus

enums.h
(full_credit only, enums.cpp not required)

```cpp
#ifndef __ENUMS_H
#define __ENUMS_H

#include <vector>
#include <string>
using namespace std;

// To convert an enumerated value to a string,
// subscript its vector, e.g., given
//      Age age = Age::teen;
// "ages[age]" will return the string version

enum Age {children, teen, adult, restricted};
const vector<string> ages =
    {"children", "teen", "adult", "restricted"};

enum Genre {fiction, nonfiction, selfhelp, performance};
const vector<string> genres =
    {"fiction", "nonfiction", "self help", "performance"};

enum Media {book, periodical, newspaper, audio, video};
const vector<string> medias =
    {"book", "periodical", "newspaper", "audio", "video"};

#endif
```

Here’s one way to approach the enums. The vectors are the “to_string”, though I didn’t use that more obvious name. :-(

You could have instead defined the enums with helper methods e.g., age_to_string, in the Publication class.

Homework #4 Sample Solutions

Full Credit and Bonus

age.h
(full_credit_mvc & bonus, no age.cpp needed)

```
#ifndef __AGE_H
#define __AGE_H 201609
#include "string"

class Age {
  public:
    Age(int val) : value(val) { }

    static const int children = 0;
    static const int teen = 1;
    static const int adult = 2;
    static const int restricted = 3;
    static const int num_ages = 4;

    string to_string() {
      switch(value) {
        case(children):return "children";
        case(teen):return "teen";
        case(adult):return "adult";
        case(restricted):return "restricted";
        default: return "UNKNOWN";
      }
    }
  private:
    int value;
};
#endif
```

Note that no .cpp file is required, since no definitions are lacking.

Regression tests are of course provided (see next slide).

Or you could use typedef, #define, or your favorite editor to define Age as a string – but solid data validation is required for this approach!

Homework #4 Sample Solutions

Full Credit and Bonus

genre.h
(no genre.cpp needed)

```
#ifndef __GENRE_H
#define __GENRE_H 201609
#include "string"

class Genre {
  public:
    Genre(int val) : value(val) { }

    static const int fiction = 0;
    static const int nonfiction = 1;
    static const int selfhelp = 2;
    static const int performance = 3;
    static const int num_genres = 4;

    string to_string() {
      switch(value) {
        case(fiction):return "fiction";
        case(nonfiction):return "nonfiction";
        case(selfhelp):return "selfhelp";
        case(performance):return "performance";
        default: return "UNKNOWN";
      }
    }
  private:
    int value;
};
#endif
```

test_genre.cpp

```
#include "genre.h"
#include <iostream>
using namespace std;

int main() {
  bool passed = true;

  Genre m1(Genre::performance);
  if (m1.to_string() != "performance") {
    passed = false;
    cerr << "performance failed: got "
         << m1.to_string() << endl;
    return 1;
  }

  return 0;
}
```

test_genre.cpp is above, and test_media.cpp and test_age.cpp are similar.

genre.h is to the left, and media.h is similar.

Homework #4 Sample Solutions

Full Credit and Bonus

patron.cpp
(bonus only)

```cpp
#include "patron.h"

string Patron::to_string() {
   return name + " (" + number + ")";
}

string Patron::get_patron_name() {return name;}
string Patron::get_patron_phone_number(){
   return number;
}
```

patron.h
(bonus only)

```cpp
#ifndef __PATRON_H
#define __PATRON_H 201609
#include <iostream>
#include <string>

using namespace std;

class Patron {
  public:
    Patron(string patron_name, string patron_phone_number)
      : name(patron_name), number(patron_phone_number) {}
    Patron() : name("unknown"), number("unknown") {}

    string to_string();

    string get_patron_name();
    string get_patron_phone_number();

  private:
    string name;
    string number;
};
#endif
```

The bonus version uses this Patron class.

The full_credit versions store name and phone directly in the Publication class.

Homework #4 Sample Solutions

Full Credit and Bonus

test_patron.cpp
(bonus only)

```
#include "patron.h"

int main() {
  string name = "Professor Rice";
  string number = "817-555-1212";

  Patron p(name, number);

  if (p.get_patron_name() != name) {
    cerr << "Name was '" << p.get_patron_name() <<
          "' should be '" << name << "'" << endl;
    return 1;
  }

  if (p.get_patron_phone_number() != number) {
    cerr << "Number was '" << p.get_patron_phone_number()  <<
          "' should be '" << number << "'" << endl;
    return 1;
  }

  string expected = "Professor Rice (817-555-1212)";
  if (p.to_string() != expected) {
    cerr << "to_string was '" << p.to_string() <<
            "' but should be '" << expected << endl;
    return 1;
  }

  return 0;
}
```

Homework #4 Sample Solutions

Full Credit and Bonus

publication.h
(bonus version using Patron)

```
#ifndef __PUBLICATION_H
#define __PUBLICATION_H 201609
#include "patron.h"
#include "media.h"
#include "genre.h"
#include "age.h"
#include <iostream>
#include <string>
using namespace std;

class Publication {
  public:
    Publication(string p_title,
                string p_author,
                string p_copyright,
                Genre p_genre,
                Media p_media,
                Age p_target_age,
                string p_isbn) :

                title(p_title),
                author(p_author),
                copyright(p_copyright),
                genre(p_genre),
                media(p_media),
                target_age(p_target_age),
                isbn(p_isbn),
                patron(Patron()),
                checked_out(false) { }
```

```
    bool is_checked_out();

    void check_out(Patron& patron);
    void check_in();

    string to_string();

    // Thrown on check_in if publication
    //   isn't checked out
    // or on cheeck_out if publication
    //   is already checked out
    class Invalid_transaction
        : public exception { };

  private:
    string title;
    string author;
    string copyright;
    Genre genre;
    Media media;
    Age target_age;
    string isbn;
    Patron patron;
    bool checked_out;
};
```

Homework #4 Sample Solutions

Full Credit and Bonus

publication.cpp
(bonus version using Patron class)

```
#include "publication.h"
#include <string>
#include <iostream>
using namespace std;

bool Publication::is_checked_out() {return checked_out;}
void Publication::check_out(Patron& p_patron) {
  if (checked_out) throw Invalid_transaction();
  else {
    patron = p_patron;
    checked_out = true;
  }
}

void Publication::check_in() {
  if (checked_out) checked_out = false;
  else throw Invalid_transaction();
}

string Publication::to_string() {
  string pub = "\"" + title + "\"" + " by " + author + ", " + copyright +
    " (" + target_age.to_string() + " " + genre.to_string() + " "
        + media.to_string() + ") " + "ISBN: " + isbn;
  if (checked_out) {
     pub += "\nChecked out to " + patron.get_patron_name() +
            " (" + patron.get_patron_phone_number() + ")";
  }
  return pub;
}
```

Homework #4 Sample Solutions

Full Credit and Bonus

library.h
(bonus version using Patron)

```
#ifndef __LIBRARY_H
#define __LIBRARY_H 201609
#include "patron.h"
#include "publication.h"
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Library {
  public:
    void add_publication(Publication pub);
    void add_patron(Patron pat);

    void check_out(int publication_index, int patron_index);
    void check_in(int publication_index);

    string publication_to_string(int publication_index);
    string patron_to_string(int patron_index);

    int number_of_publications();
    int number_of_patrons();

    void easter_egg();
  private:
    vector<Publication> publications;
    vector<Patron> patrons;
};
#endif
```

Homework #4 Sample Solutions

Full Credit and Bonus

library.cpp
(bonus version, 1 of 2)

```cpp
#include "library.h"

void Library::add_publication(Publication pub) {
  publications.push_back(pub);
}

void Library::add_patron(Patron pat) {
  patrons.push_back(pat);
}

void Library::check_out(int publication_index, int patron_index) {
  publications[publication_index].check_out(patrons[patron_index]);
}

void Library::check_in(int publication_index) {
  publications[publication_index].check_in();
}

string Library::publication_to_string(int publication_index) {
  return publications[publication_index].to_string();
}

string Library::patron_to_string(int patron_index) {
  return patrons[patron_index].to_string();
}
```

Homework #4 Sample Solutions

Full Credit and Bonus

library.cpp
(bonus version, 2 of 2)

```
int Library::number_of_publications() {
  return publications.size();
}

int Library::number_of_patrons() {
  return patrons.size();
}

void Library::easter_egg() {
 add_publication(Publication("The Firm", "John Grisham", "1991",
       Genre::fiction, Media::book, Age::adult, "0440245923"));
 add_publication(Publication("Foundation", "Isaac Asimov", "1942",
        Genre::fiction, Media::book, Age::adult, "0385177259"));
 add_publication(Publication("Foundation and Empire", "Isaac Asimov", "1943",
        Genre::fiction, Media::book, Age::adult, "0385177259"));
 add_publication(Publication("Second Foundation", "Isaac Asimov", "1944",
        Genre::fiction, Media::book, Age::adult, "0385177259"));
 add_publication(Publication("War of the Worlds", "Jeff Wayne", "1977",
        Genre::performance, Media::audio, Age::teen, "9780711969148"));
 add_publication(Publication("Willy Wonka and the Chocolate Factory", "Roald
Dahl", "1971",
        Genre::performance, Media::video, Age::children, "0142410314"));
 add_patron(Patron("Larry", "817-555-1111"));
 add_patron(Patron("Curly", "817-555-2222"));
 add_patron(Patron("Moe", "817-555-3333"));
}
```

Homework #4 Sample Solutions

Full Credit and Bonus

test_publication.cpp
(bonus version)

```
#include "publication.h"
#include "patron.h"

int main() {
  string expected = "\"The Firm\" by John
Grisham, 1991 (adult fiction book) ISBN:
0440245923";
  string expected_co = "\"The Firm\" by
John Grisham, 1991 (adult fiction book)
ISBN: 0440245923\nChecked out to Professor
Rice (817-555-1212)";

  string name = "Professor Rice";
  string number = "817-555-1212";
  Patron patron(name, number);

  // Test constructor
  Publication p("The Firm", "John Grisham",
     "1991", Genre::fiction, Media::book,
     Age::adult, "0440245923"
  );

  if (p.to_string() != expected) {
    cerr << "Constructor fail: got '"
         << p.to_string()
         << "' but expecting '"
         << expected << "'" << endl;
    return 1;
  }
```

```
  // Test check_out(patron)
  p.check_out(patron);
  if (!p.is_checked_out()) {
    cerr << "is_checked_out() reported
false after check_out(patron)" << endl;
    return 1;   }
  if (p.to_string() != expected_co) {
    cerr << "Check out fail: got '"
         << p.to_string()
         << "' but expecting '"
         << expected_co << "'" << endl;
    return 1;   }

  // Test check_in(patron)
  p.check_in();

  if (p.is_checked_out()) {
    cerr << "is_checked_out() reported true
after check_in()" << endl;
    return 1;   }

  if (p.to_string() != expected) {
    cerr << "Check in fail: got '"
         << publication.to_string()
         << "' but expecting '"
         << expected << "'" << endl;
    return 1;   }
  Return 0;
}
```

Homework #4 Sample Solutions

Full Credit and Bonus

controller.h
(full_credit_mvc
and bonus version)

```
#ifndef __CONTROLLER_H
#define __CONTROLLER_H 201609

#include "library.h"
#include "view.h"

class Controller {
  public:
    Controller (Library& lib)
    : library(lib), view(View(library)) { }
    void cli();
    void execute_cmd(int cmd);
  private:
    Library& library;
    View view;
};
#endif
```

view.h
(bonus version)

```
#ifndef __VIEW_H
#define __VIEW_H 201609

#include "library.h"

class View {
  public:
    View(Library& lib) : library(lib) { }
    void show_menu();
    void list_publications();
    void list_patrons();
    void help();
  private:
    Library& library;
};
#endif
```

This should look very familiar...

main.cpp
(full_credit_mvc
and bonus version)

```
#include "controller.h"
#include "library.h"

int main() {
  Library library;
  Controller controller(library);
  controller.cli();
}
```

The UML model only hinted at MVC, but the requirements (and particularly the strong hint that a GUI was coming in the next sprint) fairly *screamed* MVC.

Homework #4 Sample Solutions

Full Credit and Bonus

view.cpp
(bonus version, 1 of 3)

```
void View::show_menu() {
  string menu = R"(
=================================
CSE1325 Library Management System
=================================

Publications
------------
(1) Add publication
(2) List all publications
(3) Check out publication
(4) Check in publication

Patrons
=======
(5) Add patron
(6) List all patrons

Utility
-------
(9) Help
(0) Exit

)";

  cout << menu;
}
```

Raw strings permit typing virtually ANY text without escape sequences (e.g., \”), and are great for typing large blocks of text such as menus and help.

The raw string starts with
    R”(
and ends with
)”

This minimizes the number of cout uses, making transition to a GUI much easier!

Homework #4 Sample Solutions

Full Credit and Bonus

view.cpp
(bonus version, 2 of 3)

```
void View::list_publications() {
  string header = R"(
----------------------------
List of Library Publications
----------------------------
)";
  cout << header;
  for (int i=0; i<library.number_of_publications(); ++i) {
    cout << i << ") " << library.publication_to_string(i) << endl;
  }
}


void View::list_patrons() {
  string header = R"(
-----------------------
List of Beloved Patrons
-----------------------
)";
  cout << header;
  for (int i=0; i<library.number_of_patrons(); ++i) {
    cout << i << ") " << library.patron_to_string(i) << endl;
  }
}
```

Homework #4 Sample Solutions

Full Credit and Bonus

view.cpp
(bonus version, 3 of 3)

```
void View::help() {
  string helptext = R"(
The LMS tracks publication assets for a library, including those who
check out and return those publications.

(1) Add publication - This allows the librarian to enter data
    associated with a new publication.
(2) List all publications - All data known about each publication
    in the library is listed.
(3) Check out publication - Enter the data for patrons who check out
    a publication, and mark that publication as checked out.
(4) Check in publication - Select a publication and mark it as checked in.
(9) Help - Print this text.
(0) Exit - Exit the program. WARNING: The current version does NOT
    save any entered data. This feature will be added in the "next version".

Use the '99' command to pre-populate test data.
  )";
  cout << helptext << endl;
}
```

Homework #4 Sample Solutions

Full Credit and Bonus

controller.cpp
(bonus version, 1 of 4)

```
#include "controller.h"
#include "view.h"
#include "library.h"
#include "publication.h"
#include "patron.h"
#include "genre.h"
#include "media.h"
#include "age.h"
#include <iostream>
#include <string>

using namespace std;

void Controller::cli() {
  int cmd = -1;
  while (cmd != 0) {
    view.show_menu();
    cout << "Command? ";
    cin >> cmd;
    cin.ignore(); // consume \n
    execute_cmd(cmd);
  }
}
```

This should be a standard template by now

Homework #4 Sample Solutions

Full Credit and Bonus

controller.cpp
(bonus version, 2 of 4)

```
void Controller::execute_cmd(int cmd) {
  if (cmd == 1) { // Add publication
    string title, author, copyright, isbn;
    int genre, media, age;

    cout << "Title? ";    getline(cin, title);

    cout << "Author? ";    getline(cin, author);

    cout << "Copyright date? ";    getline(cin, copyright);

    for (int i = 0; i < Genre::num_genres; ++i)
      cout << "  " << i << ") " << Genre(i).to_string() << endl;
    cout << "Genre? ";    cin >> genre;    cin.ignore(); // consume \n

    for (int i = 0; i < Media::num_media; ++i)
      cout << "  " << i << ") " << Media(i).to_string() << endl;
    cout << "Media? ";    cin >> media;    cin.ignore(); // consume \n

    for (int i = 0; i < Age::num_ages; ++i)
      cout << "  " << i << ") " << Age(i).to_string() << endl;
    cout << "Age? ";    cin >> age;    cin.ignore(); // consume \n

    cout << "ISBN? ";    getline(cin, isbn);

    library.add_publication(Publication(title, author,
            copyright, genre, media, age, isbn));
```

Note the template for Genre, Media, and Age - list the options and then accept an int. Data validation is needed, though.

Homework #4 Sample Solutions

Full Credit and Bonus

```
} else if (cmd == 2) { // List all publications
   view.list_publications();

} else if (cmd == 3) { // Check out publication
   int pub, pat;

   view.list_publications();
   cout << "Publication number? ";
   cin >> pub;
   cin.ignore(); // consume \n

   view.list_patrons();
   cout << "Patron number? ";
   cin >> pat;
   cin.ignore(); // consume \n

   library.check_out(pub, pat);

} else if (cmd == 4) { // Check in publication
   int pub;
   view.list_publications();
   cout << "Publication number? ";
   cin >> pub;
   cin.ignore(); // consume \n

   library.check_in(pub);
```

controller.cpp
(bonus version, 3 of 4)

Homework #4 Sample Solutions

Full Credit and Bonus

controller.cpp
(bonus version, 4 of 4)

```
  } else if (cmd == 5) { // Add patron
     string name, number;

     cout << "Name? ";
     getline(cin, name);
     cout << "Phone number? ";
     getline(cin, number);
     library.add_patron(Patron(name, number));

  } else if (cmd == 6) { // List all patrons
     view.list_patrons();

  } else if (cmd == 9) { // Help
     view.help();
  } else if (cmd == 0) { // Exit
  } else if (cmd == 99) { // Easter Egg
    library.easter_egg();
  } else {
    cerr << "**** Invalid command - type 9 for help" << endl << endl;
  }
}
```

Homework #4 Sample Solutions

Extreme Bonus

```
#ifndef __NEWSPAPER_H
#define __NEWSPAPER_H 201609
#include "publication.h"
#include "patron.h"
#include "media.h"
#include "genre.h"
#include "age.h"
#include <iostream>
#include <string>
using namespace std;

class Newspaper : public Publication {
  public:
    Newspaper (string p_title,
          string p_author,
          string p_copyright,
          Genre p_genre,
          Media p_media,
          Age p_target_age,
          string p_isbn,
          string p_city)
        : city(p_city),
          Publication(p_title, p_author, p_copyright,
                      p_genre, p_media, p_target_age, p_isbn) { }
    virtual string to_string(); // enable polymorphic behaviors
    friend ostream& operator<<(ostream& os, Publication& p);
  protected:
    string city;
};
#endif
```

newspaper.h

Here's just a hint at the extreme bonus. Additional code is obviously needed.

We'll cover inheritance today.

Redefining operators such as << (which wasn't strictly required) is covered after GUIs.

Homework #4 Sample Solutions

Extreme Bonus

newspaper.cpp

```
#include "newspaper.h"
#include <string>
#include <iostream>

using namespace std;

string Newspaper::to_string() {
  string pub = "\"" + title + "\"" + " by " + author + ", " + copyright
        + " (" + target_age.to_string() + " " + genre.to_string() + " "
        + city + ") "
        + media.to_string()
        + " ISBN: " + isbn;
  if (checked_out) {
     pub += "\nChecked out to " + patron.get_patron_name() +
            " (" + patron.get_patron_phone_number() + ")";
  }
  return pub;
}

ostream& operator<<(ostream& os, Newspaper& p) {
    os << p.to_string();
    return os;
}
```

to_string is called *polymorphically*, e.g., if a publication object "is a" newspaper, this one is actually called.

We simply define the "<<" operator to output the to_string of this object to whatever stream is provided (e.g., cout).