# Exam-Style Practice Problems

### Problem 1 (9 points)

For the following three parts, assume that all required C++ libraries have been `#included` and that the `using namespace std;` statement is present.

Consider the following struct:

```
struct Movie {
  string title;
  int length; // minutes
};
```

Now, let's create a Movie in our main function and initialize it.

```
int main() {
  Movie m;
  Movie_init(&m, "Star Wars", 121);
}
```

**1a) (3 points)** Write the `Movie_init` initializer function. You should also fill in the first parameter in accordance with how the function is used in `main` above. Use assert to check the REQUIRES clause.

```
// REQUIRES: 'length_in' >= 0
// MODIFIES: the movie
// EFFECTS:  Initializes the movie with the given title and length
void Movie_init(                                    ,
                const string &title_in, int length_in) {
```

```
}
```

**1b) (3 points)** Suppose the following `Movie_print` function is defined.

```
// MODIFIES: standard output
// EFFECTS:  Prints out the title of the movie, a space, and the length to
//           standard output. Does not print a newline.
void Movie_print(const Movie *movie);
```

Fill in the line of code below to print out the Movie that has been declared in `main`.

```
int main() {
  Movie m;
  Movie_init(&m, "Star Wars", 121);
```

```

```

```
}
```

**1c) (3 points)** Now, consider a `Library` struct that contains an array of `Movies`. The `numMovies` member indicates the number of valid `Movie` objects in the `movies` array.

```
const int MAX_NUM_MOVIES = 10;

struct Library {
  Movie movies[MAX_NUM_MOVIES];
  int numMovies;
};
```

Write the following function, which prints out the title and length of each `Movie` in a `Library`. Your implementation must respect the interface for the `Movie` ADT. You do not have to check the REQUIRES clause.

```
// REQUIRES: The numMovies member is >= 0, and the first numMovies items in
//           the movies array are valid.
// MODIFIES: standard output
// EFFECTS:  Prints information about the movies in the Library to standard
//           out. Each movie is followed by a newline, including the last.
// NOTE: Movies are printed in the same order as they appear in the array.
void Library_print(const Library *lib_ptr) {
```

```

```

```
}
```