[**Open notes**] Show all of your work clearly in the space provided or on the additional page at the end of the exam. If the additional page is used, clearly identify to which exam question it is related. Be sure to **read each problem carefully**. You should answer all 4 questions. Note that the exam is double sided.

1. (15 points) Briefly, in your own words, describe three advantages of using abstract data types.

2. (10 points) Briefly, in your own words, explain the term **memory leak**. Use examples when appropriate.

3. (20 points) Suppose that the Rational class that we developed this quarter is modified so that the default constructor generates a Rational object that contains a random value. Make use of the Rational class (given on the last page of the exam for reference) to write the following educational "game".

Write a program that will generate two rational numbers (with random values) and ask the user to enter the result of adding, subtracting, multiplying, and dividing the numbers together. The program should indicate whether the user answered correctly and give the correct answer for any incorrect answers. Below is a sample user interaction where the text between | marks is entered by the user:

```
Please enter the answers to the following:
1/3 + 5/7 = |22/21|
Correct!
1/3 - 5/7 = |8/21|
Incorrect, 1/3 - 5/7 = -8/21
Would you like to play again? (y or Y for yes) |y|
Please enter the answers to the following:
2/1 + -3/8 = |13/8|
Correct!
2/1 - -3/8 = |19/8|
Correct!
Would you like to play again? (y or Y for yes) |n|
Bye!
```

Write all of the commands necessary to compile your answer as a stand-alone file.

3. cont…

4. Recall that in lab 2 you implemented an Album class that produced a `.tag` file like the one below:

```
TITLE=Disk 8
YEAR=1996
FILENAME1=70.mp3
FILENAME2=71.mp3
TRACK1=The Seventy Weeks and the Grea
TRACK2=Seventy-one Portraits in Jazz
ARTIST1=Devos, Richard M.
ARTIST2=Bennett, William Gordon
LENGTH1=1.342
LENGTH2=1.59
```

Storing the actual MP3 objects in a container within the Album class takes a lot of memory. One way around this is to create multiple containers to hold each piece of information associated with a track on the album. This results in having to manage multiple containers and can be a pain. An alternate approach would be to create a Track class which just holds the information associated with a track that is of interest to the Album class. Listed below is definition of a Track class.

```
class Track {
public:
  Track();
  Track(const MP3& song);
  Track(const Track& org);
  ~Track();
  Track& operator=(const Track& rhs);
  bool operator==(const Track& rhs) const;
  bool operator<(const Track& rhs) const;
  string getFilename() const;
  string getTitle() const;
  string getArtist() const;
  double getLength() const;
```

```
private:
  string filename;
  string title;
  string artist;
  double length;
};
```

(a) (10 points) Implement the getTitle member function.

(b) (10 points) Implement the assignment operator.

(c) (15 points) Implement the less than operator so that if listOfTrackObjects is a std::list<Track>, then listOfTrackObjects.sort(); results in a list of Track objects from shortest to longest in length.

(d) (20 points) On page 4, add the appropriate function(s) in order to allow inserting a Track object into an output stream (e.g., cout << trkObj;), and implement the function(s) here.

The output should have the following format:

```
70.mp3: "The Seventy Weeks and the Grea" by Devos, Richard M. [1.342 secs]
```

Reference for problem **3.**

```
#include <iostream>
using namespace std;

class Rational {
public:
  Rational(); // Now creates a rational number with a random value
  Rational(const Rational& org);
  ~Rational();
  Rational& operator=(const Rational& rhs);
  Rational operator+(const Rational& rhs) const;
  Rational operator-(const Rational& rhs) const;
  Rational operator/(const Rational& rhs) const;
  Rational operator*(const Rational& rhs) const;
  void display() const;
  bool extract(istream& is);
  Rational& operator+=(const Rational& rhs);
  Rational& operator-=(const Rational& rhs);
  Rational& operator/=(const Rational& rhs);
  Rational& operator*=(const Rational& rhs);
  bool operator!=(const Rational& rhs) const;
  bool operator==(const Rational& rhs) const;
  bool operator<(const Rational& rhs) const;
private:
  reduce();
  long int num;
  unsigned long int den;
};

Rational operator+(long int lhs, const Rational& rhs);
Rational operator-(long int lhs, const Rational& rhs);
Rational operator/(long int lhs, const Rational& rhs);
Rational operator*(long int lhs, const Rational& rhs);
ostream& operator<<(ostream& os, const Rational& rhs);
istream& operator>>(istream& is, const Rational& rhs);
```