makefile

```
################################################################
# Program:
#     Lesson 01, SET
#     Brother Helfrich, CS235
# Author:
#     David Lambertson and Derek Calkins
# Summary:
#     this makefile takes our different files and makes a TAR and a.out.
#     The a.out file lets the user run the program and test the Set Class
#     written by us. It has four tests and lets them play Go Fish using the
#     goFish.cpp written by us also. The tar lets us submit our assignment.
################################################################

################################################################
# The main rule
################################################################
a.out: set.h lesson01.o goFish.o card.o
       g++ -g -o a.out lesson01.o goFish.o card.o
       tar -cf lesson01.tar *.h *.cpp makefile

################################################################
# The individual components
#      lesson01.o     : the driver program
#      goFish.o       : the logic for the goFish game
#      card.o         : a single playing card
################################################################
lesson01.o: set.h goFish.h lesson01.cpp
       g++ -g -c lesson01.cpp

goFish.o: set.h goFish.h goFish.cpp card.h
       g++ -g -c goFish.cpp

card.o: card.h card.cpp
       g++ -g -c card.cpp
```

Commented [HJ1]: How long did it take for you to complete this assignment

goFish.h

```
/***********************************************************************
* Header:
*    Go Fish
* Summary:
*    This will contain just the prototype for the goFish() function
* Author
*    <your names here>
***********************************************************************/

#ifndef GO_FISH_H
#define GO_FISH_H

/*********************************************
 * GO FISH
 * Play the game of "Go Fish"
 *********************************************/
void goFish();


#endif // GO_FISH_H
```

union.h

```
//Hi this is a prototype
template <class T>
Set<T> operator && (Set<T> rhs)
{
   iSet1 = 0;
   iSet2 = 0;
```

```
   Set<T> setReturn;
   while ( iSet1 < numItems || iSet < rhs.size())
   {
      if (iSet1 == numItems)
         return SetReturn;
      else if ( iSet2 == rhs.size())
         return setReturn;
      else if ( data[iSet1] == rhs.getData(iSet2))
      {
         setReturn.insert(data[iSet1]);
         iSet1++;
         iset2++;
      }
      else if (data[iSet1] < rhs.getData(iSet2))
         iSet1++;
      else
         iSet2++;
   }

   return setReturn;
}
```

Commented [HJ2]: Interesting. This needs to be in set.h

card.h

```
/***********************************************************************
* Header File
*    This is the header file for a "Go Fish" card
* Summary:
*     Playing cards for the children's version of Go Fish
* Author:
*    Br. Helfrich
***********************************************************************/

#ifndef CARD_H
#define CARD_H

#include <iostream>       // for IFSTREAM and OFSTREAM
#include <cassert>        // for ASSERT in the constructors

#define INDEX_FIRST 1
#define INDEX_LAST  6
#define INVALID     0

/*****************************************
 * CARD
 * Card class for the children's version
 * of Go Fish. The text of the cards is
 * described at the top of card.cpp
 *****************************************/
class Card
{
  public:
   // various constructors
   Card()                  : value(INVALID)   { assert(validate()); }
   Card(const Card & rhs)  : value(rhs.value) { assert(validate()); }
   Card(const char * rhs)  : value(INVALID)   { *this = rhs;        }

   bool isInvalid() const { return value == INVALID; }

   // insertion and extraction operators
   friend std::ostream & operator << (std::ostream & out, const Card & card);
   friend std::istream & operator >> (std::istream & in,        Card & card);

   // assignment
   Card & operator = (const Card & rhs);     // copy one card to another
   Card & operator = (const char * rhs);     // assign the string to the card

   // Absolute and relative comparision... comparing cards
   bool operator == (const Card & rhs) const { return value == rhs.value; }
   bool operator != (const Card & rhs) const { return value != rhs.value; }
   bool operator >= (const Card & rhs) const { return value >= rhs.value; }
   bool operator >  (const Card & rhs) const { return value >  rhs.value; }
   bool operator <= (const Card & rhs) const { return value <= rhs.value; }
   bool operator <  (const Card & rhs) const { return value <  rhs.value; }


  private:
```

Commented [HJ3]: Unchanged.

```
   // holds the value.  Though there are 256 possible, only 52 are used
   unsigned char value;               // internal representation

   // private functions
   bool validate() const;             // are we in a valid state?
};

#endif // CARD_H
```

set.h

```
/*********************************
 *This is my(and Derek's) set.h File. DO NOT TOUCH!!
 *It implements the class Set and its iterator.
 *Please Do not copy this. thank you, Have a Nice Day.

/************************************************************************
* Program:
*    Assignment 01, set.h
*    Brother Helfrich, CS 235
* Author:
*    David Lambertson
* Summary:
*    This program is the implementation of the set class. It creates Sets and
*    all of the necessary functions to go along with them. It also implements
*    the SetIterator class to create iterators for the Sets.
*
*    Estimated:  5.0 hrs
*    Actual:     7.0 hrs
*      The most difficult part for me was getting my insert to work properly.
************************************************************************/

#ifndef SET_H
#define SET_H

#include <cassert>


template <class T>
class SetIterator;

template<class T>
class Set      //HIKE!
{
  public:
   //Default Constructor
  Set() : myCapacity(0), numItems(0), data(0x00000000) {}

   //Non-default Constructor
   Set(int capacity) throw (const char *);

   //Copy Constructor
   Set(const Set & rhs) throw (const char *) { *this = rhs; }

   //Destructor
   ~Set() {delete [] data; }


   //Checks if the Set is empty
   bool empty() const { return numItems == 00; }

   //returns the size of the Set
   int size() const {return numItems; }

   //inserts a value into the Set at the right spot.
   void insert(const T & value);

   //erases an item given by the user if found. NEEDs
   void erase(SetIterator<T> item);

   // overloaded assignment operator
   Set<T> & operator = (const Set<T>& rhs);

   //returns an iterator for the beginning of the Set
   SetIterator<T> begin() { return SetIterator<T>(data); }

   //returns an iterator for the end of the Set
   SetIterator<T> end() { return SetIterator<T>(data + numItems); }
```

Commented [HJ4]: This was your total component?

Commented [HJ5]: You need a comment block …
```
/*************************
 *
 *************************/
```
… before each class definition

Commented [HJ6]:

Commented [HJ7]: Could throw.

Commented [HJ8]: Could throw.

```
   // finds if a given item is in the Set
   SetIterator<T> find(const T & value);

   //lets the us access the data without changing it.
   T getData(int spot) const { return data[spot]; }

   //overloaded union operator
   Set<T> operator || (Set<T> rhs);

   //overloaded intersection operator
   Set<T> operator && (Set<T> rhs);


  private:
   int numItems;
   int myCapacity;
   T * data;

   //implements the reallocation of data if needed;
   void regrowth();

   // locates a value within the Set if it exists
   int locate(const T & value);

};

/*******************************
 * This is the class for the iterator
 * for set which is shown above.
 *******************************/
template <class T>
class SetIterator
{
  public:

   //Default Constructor
  SetIterator() : p(0x00000000) {}

   //Non Default Constructor
  SetIterator(T * p) : p(p) {}

   //copy Constructor
   SetIterator(const SetIterator & rhs) { this->p = rhs.p; }

   //overloaded assignment operator
   SetIterator & operator = (const SetIterator & rhs)
   {
      this->p = rhs.p;
      return *this;
   }

   //overloaded not equal operator
   bool operator != (const SetIterator & rhs) const
   {
      return rhs.p != this->p;
   }

   //overloaded by-reference operator
   T & operator * ()
   {
      return *p;
   }

   //overloaded prefix add one operator
   SetIterator <T> & operator ++ ()
   {
      p++;
      return *this;
   }

   //overloaded postfix add one operator
   SetIterator <T> operator++ (int postfix)
   {
      SetIterator tmp(*this);
      p++;
      return tmp;
   }
```

Commented [HJ9]: Should be private; only private member variables know about your array.

Commented [HJ10]: Should be a const method; *this does not change.

Commented [HJ11]: Should be a const method; *this does not change.

Commented [HJ12]: Good.

```
  private:
   T * p; //the pointer for the iterator

};

/************************
 * This is the implementation of
 * the non Default Constructor
 ************************/
template <class T>
Set<T> :: Set(int capacity) throw (const char *)
{
   assert (capacity >= 0);

   if (capacity == 0)
   {
      this->myCapacity = this->numItems = 0;
      this->data = 0x00000000;
      return;
   }

    try
   {
      this->data = new T[capacity];
   }

   catch(...)
   {
      throw "Error: Unable to allocate buffer";
   }

   this->myCapacity = capacity;
   this->numItems = 0;
}


/************************
 * Implementation of the overloaded assignment
 * operator, will be used within copy.
 ************************/
template <class T>
Set<T> & Set<T> :: operator = (const Set<T>& rhs)
{
   assert(rhs.myCapacity >=0);

   if (rhs.myCapacity == 0)
   {
      this->myCapacity = this->numItems = 0;
      this->data = 0x00000000;
      return *this;
   }

   try
   {
      this->data = new T[rhs.myCapacity];
   }

   catch(...)
   {
      throw "Error: Unable to allocate buffer";
   }

   this->myCapacity = rhs.myCapacity;
   this->numItems = rhs.numItems;
   for (int i = 0; i < numItems; i++)
      this->data[i] = rhs.data[i];

   return *this;
}

/*****************************
 *INSERT (T Value)
 * this function will take the value given by
 * the user and insert it into the function.
 *****************************/
template <class T>
void Set<T> :: insert(const T & value)
{
```

Commented [HJ13]: Call resize(); not not duplicate that code.

```
   if (myCapacity == 0)                          // capacity is zero, makes it one and adds the item
   {
      myCapacity += 1;
      data = new T[myCapacity];
      data[numItems] = value;
      numItems++;
      return;
   }

   if (myCapacity == numItems)                   // if the data needs reallocating, do it here
   {
      regrowth();
   }


   int spot = locate(value);

   if (data[spot] == value)                     // if data is found, do nothing
   {
      return;
   }
   else                                         //else find where it belongs and put it there
   {

      T *newData = new T[myCapacity];

      if (value < data[spot])
      {

         int i;                                 //just like the reallocation, just in parts to
                                                //insert the new value
         for (i = 0; data[i] != data[spot]; i++) // moves the data that comes
         {                                      //before the value user wants to insert.
            newData[i] = data[i];
         }
         newData[i] = value;                    //saves the value into the Set
         for (i; data[i] != data[numItems]; i++) //saves the rest of the data left over.
         {
            newData[i+ 1] = data[i];
         }
         delete [] data;                        //delete the old data
         data = newData;                        //point the data to the newData
      }
      else                                      //saves the user value at the end of the Set
      {
         data[numItems] = value;
      }
   }

   numItems++;                                  //increment the number of items
}

/**********************
 *FIND I SHALL FIND YOU!!!
 * Lets the user find if an item
 * is in the Set. if not, it points to
 * the end of the Set.
 ************************/
template <class T>
SetIterator<T> Set<T> :: find(const T & value)
{
   int spot = locate(value);

   if ( data[spot] == value)
      return SetIterator<T> (&data[spot]);
   else
      return end();

}

/**********************
 *YOU SHALL BE DESTROYED!
 *lets the user erase a value if it
 * can be found. if not, does nothing.
 ************************/
template <class T>
void Set<T> :: erase(SetIterator<T> item)
{
   int spot = locate(*item);
```

Commented [HJ14]: Every efficient.

Commented [HJ15]: Good.

```
   if (data[spot]  == *item)
   {
      numItems--;
      for (int i = spot; i < numItems; i++)
         data[i] = data[i+1];
   }
}


/*************************
 * This lets us use the ||
 * operator in other code.
 *************************/
template <class T>
Set<T> Set<T> :: operator || (Set<T> rhs)
{
   int iSet1 = 0;
   int iSet2 = 0;
   // SetIterator<T> value;
   Set<T> setReturn;
   while ( iSet1 < numItems || iSet2 < rhs.size())
   {
      // value = rhs.find(iSet2);
      if (iSet1 == numItems)
         setReturn.insert(rhs.getData(iSet2++));

      else if (iSet2 == rhs.size())
         setReturn.insert(data[iSet1++]);

      else if (data[iSet1] == rhs.getData(iSet2))
      {
         setReturn.insert(data[iSet1]);
         iSet1++;
         iSet2++;
      }

      else if ( data[iSet1] < rhs.getData(iSet2))
         setReturn.insert(data[iSet1++]);

      else
         setReturn.insert(rhs.getData(iSet2++));
   }
   return setReturn;
}

/*************************
 * this lets us use && in other lines of
 * code when it comes to Sets.
 *************************/
template <class T>
Set<T> Set<T> :: operator && (Set<T> rhs)
{
   int iSet1 = 0;
   int iSet2 = 0;

   Set<T> setReturn;
   while ( iSet1 < numItems || iSet2 < rhs.size())
   {
      if (iSet1 == numItems)
         return setReturn;

      else if ( iSet2 == rhs.size())
         return setReturn;

      else if ( data[iSet1] == rhs.getData(iSet2))
      {
         setReturn.insert(data[iSet1]);
         iSet1++;
         iSet2++;
      }

      else if (data[iSet1] < rhs.getData(iSet2))
         iSet1++;

      else
         iSet2++;
   }

   return setReturn;
```

Commented [HJ16]: Well done.

```
}


/**********************
 * This function takes the capacity of the Set
 * and doubles and reallocates the data.
 **********************/
template <class T>
void Set <T> :: regrowth()
{
   T * newData;
   myCapacity *= 2;

   try
   {
      newData = new T[myCapacity];
   }

   catch(...)
   {
      throw "Unable to allocate a buffer for Set.";
   }

   for (int i = 0; i < numItems; i++)
   {
      newData[i] = data[i];
   }

   delete [] data;
   data = newData;
}

/************************
 * This function uses a binary search to locate
 * the given value in the Set, if it is not found
 * returns the value right below it.
 ***************************/
template<class T>
int Set<T> :: locate(const T & value)
{
   int find  = 0;
   int low = 0;
   int high = (numItems - 1);

   while ( low <= high)
   {
      find = (high+low)/2;
      if (data[find] == value)
      {
         return find;
      }
      else if (value < data[find])
      {
         high = find - 1;
      }
      else
         low = find + 1;
   }
   if (data[find] < value)
      find++;

   return find;
}

#endif
```

goFish.cpp

```
/***********************************************************************
* Program:
*    Assignment 01, Go Fish
*    Brother Helfrich, CS 235
* Author:
*    Derek Calkins
* Summary:
*    This is all the functions necessary to play Go Fish!
*    (this information is just for Derek's time)
```

```
*
*    Estimated:  10.0 hrs
*    Actual:     7.0 hrs
*      The most difficult
*************************************************************************/

#include <fstream>
#include <string>
#include "set.h"
#include "card.h"
#include "goFish.h"
using namespace std;

/*************************************************************************
 * GO FISH
 * This function reads in the file into a set, starts the game, and then
 * takes the input from the user and removes it if it is found, and
 * doesn't do anything if it is not found. Then displays what is left
 * over in the hand after the 5 rounds are played.
 *************************************************************************/
void goFish()
{
   string fileName = "/home/cs235/lesson01/hand.txt";

   ifstream fin(fileName.c_str());
   if (fin.fail())
   {
      cout << "NOT WORKING!!!";
   }

   Set <Card> hand;
   Card input;

   //reads file into hand
   while (fin >> input)
   {
      hand.insert(input);
   }

   fin.close();

   int num = 0;

   cout << "We will play 5 rounds of Go Fish.  Guess the card in the hand\n";
   for (int i = 1; i <= 5; i++)
   {
      cout << "round " << i << ": ";
      cin  >> input;

      //if the input cannot be found, Go Fish, otherwise match and erase. :)
      if(hand.find(input) != hand.end())
      {
         num++;
         hand.erase(hand.find(input));
         cout << "\tYou got a match!\n";
      }
      else
         cout << "\tGo Fish!\n";
   }

   // finds how many items are in the
   int numItems = hand.size();

   cout << "You have " << num << " matches!\n";
   cout << "The remaining cards: ";

   SetIterator<Card> it;

   for (it = hand.begin(); it != hand.end(); it++)
   {
      //for already finding at least one more item in hand
      numItems--;

      //if there is something left put comma
      if (numItems)
      {
         cout << *it << ", ";
      }
```

Commented [HJ17]: Should be a separate function

```
      //new line when last item in hand
      else
         cout << *it << endl;
   }
}
```

Commented [HJ18]: Shoujl be a separate f unction.

lesson01.cpp

```
/**********************************************************************
* Program:
*    Lesson 01, Set
*    Brother Helfrich, CS 235
* Author:
*    Br. Helfrich
* Summary:
*    This is a driver program to exercise the Set class.  When you
*    submit your program, this should not be changed in any way.  That being
*    said, you may need to modify this once or twice to get it to work.
**********************************************************************/

#include <iostream>       // for CIN and COUT
#include <string>         // because testIterate() uses a Set of string
#include "set.h"          // your Set class needs to be in set.h
#include "goFish.h"       // your goFish() function needs to be defined here
using namespace std;

// prototypes for our four test functions
void testSimple();
void testFill();
void testFind();
void testUnionIntersection();

// To get your program to compile, you might need to comment out a few
// of these. The idea is to help you avoid too many compile errors at once.
// I suggest first commenting out all of these tests, then try to use only
// TEST1.  Then, when TEST1 works, try TEST2 and so on.
#define TEST1   // for testSimple()
#define TEST2   // for testFill()
#define TEST3   // for testFind()
#define TEST4   // for testUnionItersection()

/**********************************************************************
 * MAIN
 * This is just a simple menu to launch a collection of tests
 **********************************************************************/
int main()
{
   // menu
   cout << "Select the test you want to run:\n";
   cout << "\t0. Go Fish!\n";
   cout << "\t1. Just create and destroy a Set.\n";
   cout << "\t2. The above plus fill and iterate through the Set.\n";
   cout << "\t3. The above plus find if an item is in the Set.\n";
   cout << "\t4. The above plus union and intersection.\n";

   // select
   int choice;
   cout << "> ";
   cin  >> choice;
   switch (choice)
   {
      case 0:
         goFish();
         break;
      case 1:
         testSimple();
         cout << "Test 1 complete\n";
         break;
      case 2:
         testFill();
         cout << "Test 2 complete\n";
         break;
      case 3:
         testFind();
         cout << "Test 3 complete\n";
         break;
      case 4:
         testUnionIntersection();
```

```
         cout << "Test 4 complete\n";
         break;
      default:
         cout << "Unrecognized command, exiting...\n";
   }

   return 0;
}

/****************************************
 * TEST SIMPLE
 * Very simple test for a set: create and destroy
 ****************************************/
void testSimple()
{
#ifdef TEST1
   // Test1: bool Set with default constructor
   cout << "Create a bool Set using default constructor\n";
   Set <bool> s1;
   cout << "\tSize:  " << s1.size() << endl;
   cout << "\tEmpty? " << (s1.empty() ? "Yes" : "No") << endl;

   {
      // Test2: double Set with non-default constructor
      cout << "Create a double Set using the non-default constructor\n";
      Set <double> s2(10 /*capacity*/);
      cout << "\tSize:  " << s2.size() << endl;
      cout << "\tEmpty? " << (s2.empty() ? "Yes" : "No") << endl;
   }
   cout << "\tDestroying the second Set\n";
#endif // TEST1
}

/****************************************
 * TEST FILL
 * This will test the following:
 *   1. Instantiating a Set object
 *   2. Filling the contents with values
 *   3. Iterate through the set to display the contents
 *   4. Destroying an object when finished
 ****************************************/
void testFill()
{
#ifdef TEST2
   // create
   cout << "Create an integer Set with the default constructor\n";
   Set <int> s;

   cout << "\tEnter numbers, type 0 when done\n";
   int number;
   do
   {
      cout << "\t> ";
      cin  >> number;
      if (number)
         s.insert(number);
   }
   while (number);

   // display how big it is
   cout << "\tSize:  " << s.size() << endl;
   cout << "\tEmpty? " << (s.empty() ? "Yes" : "No") << endl;

   // iterate through the set
   cout << "Iterate through the set and display the contents\n";
   SetIterator <int> it;
   for (it = s.begin(); it != s.end(); ++it)
      cout << "\t" << *it << endl;
#endif // TEST2
}

/****************************************
 * TEST FIND
 * This will test the following:
 *   1. Instantiating a Set object
 *   2. Filling the contents with values
 *   3. Displaying the values using an iterator
 *   4. Prompt for the existance of an item in the set and remove it
 *   5. Display the remaining items in the set
```

```
 ****************************************/
void testFind()
{
#ifdef TEST3
   // create a list
   cout << "Create a Set of strings with the default constructor.\n";
   Set <string> s;

   // fill the Set with text
   cout << "\tEnter text, type \"quit\" when done\n";
   string text;
   do
   {
      cout << "\t> ";
      cin  >> text;
      if (text != "quit")
         s.insert(text);
   }
   while (text != "quit");

   // display the contents of the Set
   cout << "Use the iterator to display the contents of the Set\n";
   SetIterator <string> it;
   for (it = s.begin(); it != s.end(); ++it)
      cout << "\t" << *it << endl;

   // look for an item in the set
   cout << "Find items in the set and delete.\n";
   cout << "\tEnter words to search for, type \"quit\" when done\n";
   cout << "\t> ";
   cin  >> text;
   do
   {
      SetIterator <string> itEmpty = s.end();
      SetIterator <string> itFind = s.find(text);
      if (itFind != itEmpty)
      {
         cout << "\tFound and removed!\n";
         s.erase(itFind);
      }
      else
         cout << "\tNot found\n";
      cout << "\t> ";
      cin  >> text;
   }
   while (text != "quit");

   // show the list again
   cout << "The remaining list after the items were removed\n";
   for (it = s.begin(); it != s.end(); ++it)
      cout << "\t" << *it<< endl;

#endif // TEST3
}

/*****************************************
 * TEST UNION INTERSECTION
 * This will test the following:
 *    1. Instantiate two Set objects and fill them
 *    2. Display the results of Union
 *    4. Display the results of Intersection
 *****************************************/
void testUnionIntersection()
{
#ifdef TEST4
   cout.setf(ios::fixed | ios::showpoint);
   cout.precision(1);

   // fill the first set with numbers
   Set <float> s1;
   cout << "First set: enter numbers, type 0.0 when done\n";
   float number;
   do
   {
      cout << "\t> ";
      cin  >> number;
      if (number != 0.0)
         s1.insert(number);
   }
```

```
   while (number != 0.0);

   // fill the second set with numbers
   Set <float> s2;
   cout << "Second set: enter numbers, type 0.0 when done\n";
   do
   {
      cout << "\t> ";
      cin  >> number;
      if (number != 0.0)
         s2.insert(number);
   }
   while (number != 0.0);

   // display union
   cout << "s1 && s2:\n";
   Set <float> sUnion(s1 && s2);
   SetIterator <float> it;
   for (it = sUnion.begin(); it != sUnion.end(); ++it)
      cout << "\t" << *it << endl;

   // display intersection
   cout << "s1 || s2:\n";
   Set <float> sIntersection(s1 || s2);
   for (it = sIntersection.begin(); it != sIntersection.end(); ++it)
      cout << "\t" << *it << endl;

#endif // TEST4
}
```

tmp.cpp

```
#include <iostream>
#include "card.h"
#include <fstream>
#include <string>
#include "../faculty/lesson01/set.h"
using namespace std;

int main()
{
   string filename;
   cout << "filename: ";
   cin  >> filename;

   ifstream fin(filename.c_str());
   if (fin.fail())
   {
      cout << "Looser!\n";
      return 1;
   }

   Set <Card> hand;
   Card c;

   while (fin >> c)
      hand.insert(c);

   fin.close();

   cout << "my cards: ";
   for (SetIterator <Card> it = v.begin(); it != v.end(); ++it)
      cout << *it << ' ';
   cout << endl;
   return 0;
}
```

card.cpp

```
/***********************************************************************
* Source File:
*    Implementation of the Card class.
* Summary:
*    This file will implement all the methods described in card.h
* Author:
*    Br. Helfrich
************************************************************************/
```

```cpp
#include <iostream>       // for IFSTREAM and OFSTREAM
#include <cassert>        // because I am paranoid
#include <string.h>       // for STRCMP
#include "card.h"         // for the class definition
using namespace std;

/************************************
 * CARD NAMES
 * The name on each card
 ************************************/
const char * CARD_NAMES[] =
{
   "-INVALID-", // 0
   "AngleFish", // 1
   "Cod",       // 2
   "Crab",      // 3
   "Dolphin",   // 4
   "SeaHorse",  // 5
   "Shark",     // 6
};

/*****************************
 * Insertion  cout << x;
 * RETURN:    ostream by reference   (so we can say (cout << x) << y;)
 * PARAMETER: ostream by reference   (we do not want a copy of cout)
 *            constant by reference  (we do not want to make a copy or change)
 *****************************/
ostream & operator << (ostream & out, const Card & card)
{
   // we better be one of the valid cards...
   assert(card.validate());

   // display the friendly name of the card
   out << CARD_NAMES[card.value];

   // return the output stream
   return out;
}

/*****************************
 * Extraction    cin >> x;
 * RETURN:    istream by reference  (so we can say (cin >> x) >> y;)
 * PARAMETER: istream by reference  (we do not want a copy of cin)
 *            by reference          (no copies but we do want to change this)
 *****************************/
istream & operator >> (istream & in, Card & card)
{
   // input comes in the form of a string
   string input;
   in >> input;

   // do the actual work
   card = input.c_str();
   assert(card.validate());

   // return the input stream
   return in;
}

/************************************
 * ASSIGNMENT
 * RETURN:    *this by reference
 * PARAMETER: a constant Card
 * METHOD:    not const
 ************************************/
Card & Card :: operator = (const Card & rhs)
{
   assert(rhs.validate());
   value = rhs.value;
   return *this;
}

/************************************
 * ASSIGNMENT
 * RETURN:    *this by reference
 * PARAMETER: a constant string representing the card
 * METHOD:    not const
 ************************************/
Card & Card :: operator = (const char * rhs)
```

```
{
   // initially we don't know the value
   value = INVALID;
   for (int i = INDEX_FIRST; value == INVALID && i <= INDEX_LAST; i++)
      if (strcmp(rhs, CARD_NAMES[i]) == 0)
         value = i;

   assert(validate());
   return *this;
}

/*****************************
 * VALIDATE
 *****************************/
bool Card :: validate() const
{
   return (value == INVALID ||
       (value >= INDEX_FIRST && value <= INDEX_LAST));
}
```

Test Bed Results

```
Test bed did not pass

cs235d.out:

Started program
   > Select the test you want to run:
   >    0. Go Fish!
   >    1. Just create and destroy a Set.
   >    2. The above plus fill and iterate through the Set.
   >    3. The above plus find if an item is in the Set.
   >    4. The above plus union and intersection.
   > > 1
   > Create a bool Set using default constructor
   >    Size:  0
   >    Empty? Yes
   > Create a double Set using the non-default constructor
   >    Size:  0
   >    Empty? Yes
   >    Destroying the second Set
   > Test 1 complete
Program terminated successfully

Started program
   > Select the test you want to run:
   >    0. Go Fish!
   >    1. Just create and destroy a Set.
   >    2. The above plus fill and iterate through the Set.
   >    3. The above plus find if an item is in the Set.
   >    4. The above plus union and intersection.
   > > 2
   > Create an integer Set with the default constructor
   >    Enter numbers, type 0 when done
   >    > 4
   >    > 4
   >    > 4
   >    > 2
   >    > 6
   >    > 3
   >    > 5
   >    > 4
   >    > 4
   >    > 1
   >    > 7
   >    > 0
   >    Size:  7
   >    Empty? No
   > Iterate through the set and display the contents
   >    1
   >    2
   >    3
   >    4
   >    5
   >    6
   >    7
   > Test 2 complete
Program terminated successfully
```

```
Started program
   > Select the test you want to run:
   >    0. Go Fish!
   >    1. Just create and destroy a Set.
   >    2. The above plus fill and iterate through the Set.
   >    3. The above plus find if an item is in the Set.
   >    4. The above plus union and intersection.
   > > 3
   > Create a Set of strings with the default constructor.
   >    Enter text, type "quit" when done
   >    > beta
   >    > alpha
   >    > alpha
   >    > epsilon
   >    > delta
   >    > theta
   >    > platypus
   >    > theta
   >    > upsilon
   >    > capybara
   >    > quit
   > Use the iterator to display the contents of the Set
   >    alpha
   >    beta
   >    capybara
   >    delta
   >    epsilon
   >    platypus
   >    theta
   >    upsilon
   > Find items in the set and delete.
   >    Enter words to search for, type "quit" when done
   >    > capybara
   >    Found and removed!
   >    > capybara
   >    Not found
   >    > tapir
   >    Not found
   >    > platypus
   >    Found and removed!
   >    > quit
   > The remaining list after the items were removed
   >    alpha
   >    beta
   >    delta
   >    epsilon
   >    theta
   >    upsilon
   > Test 3 complete
Program terminated successfully

Started program
   > Select the test you want to run:
   >    0. Go Fish!
   >    1. Just create and destroy a Set.
   >    2. The above plus fill and iterate through the Set.
   >    3. The above plus find if an item is in the Set.
   >    4. The above plus union and intersection.
   > > 4
   > First set: enter numbers, type 0.0 when done
   >    > 2
   >    > 4
   >    > 6
   >    > 8
   >    > 0
   > Second set: enter numbers, type 0.0 when done
   >    > 3
   >    > 4
   >    > 5
   >    > 6
   >    > 7
   >    > 0
   > s1 && s2:
   >    4.0
   >    6.0
   > s1 || s2:
   >    2.0
   >    3.0
```

```
   >    4.0
   >    5.0
   >    6.0
   >    7.0
   >    8.0
   > Test 4 complete
Program terminated successfully

Started program
   > Select the test you want to run:
   >    0. Go Fish!
   >    1. Just create and destroy a Set.
   >    2. The above plus fill and iterate through the Set.
   >    3. The above plus find if an item is in the Set.
   >    4. The above plus union and intersection.
   > > 0
   > We will play 5 rounds of Go Fish.  Guess the card in the hand
   > round 1: Shark
   >    You got a match!
   > round 2: Shark
   >    Go Fish!
   > round 3: Goldfish
   >    Go Fish!
   > round 4: Salmon
   >    Go Fish!
   > round 5: Cod
   >    You got a match!
   > You have 2 matches!
   > The remaining cards: AngleFish, Crab, Dolphin, SeaHorse
Program terminated successfully

No Errors
```

Grading Criteria

Criteria	Exceptional 100%	Good 90%	Acceptable 70%	Developing 50%	Missing 0%	Weight	Score
Set interface	The interfaces are perfectly specified with respect to const, pass-by-reference, etc.	lesson01.cpp compiles without modification	All of the methods in Set and SetIterator match the problem definition	Both Set and SetIterator have many of the same interfaces as the problem definition	The public methods in the Set class do not resemble the problem definition	20	18
Set Implementation	Passes all four Set testBed tests	Passes three testBed tests	Passes two testBed tests	Passes one testBed test	Program fails to compile or does not pass any testBed tests	20	20
Go Fish	The code is elegant and efficient	Passes the Go Fish testBed test	The code essentially works but with minor defects	Elements of the solution are present	The game Go Fish was not attempted	30	30
Code Quality	There is no obvious room for improvement	All the principles of encapsulation and modularization are honored	One function is written in a "backwards" way or could be improved	Two or more functions appears "thrown together."	The code appears to be written without any obvious forethought	20	16
Style	Great variable names, no errors, great comments	No obvious style errors	A few minor style errors: non-standard spacing, poor variable names, missing comments, etc.	Overly generic variable names, misleading comments, or other gross style errors	No knowledge of the BYU-I code style guidelines were demonstrated	10	9
Extra Credit	Second game 20%					20	
Total							93