```
/*
 * Date.h
 *
 *  Created on: Oct 16, 2013
 *      Author: Nathaniel Gallinger
 */

#ifndef DATE_H_
#define DATE_H_

namespace NathanielGallinger
{
  class Date
  {
  public:
    // Constructor
    Date();
    Date(int month, int day, int year);

    // Accessors and Mutators
    inline int getMonth() const;
    inline int getDay() const;
    inline int getYear() const;

    // Display function
    void display();

  private:
    // Private data members
    int month;
    int day;
    int year;

    // Check for valid date
    int checkDate(int month, int day, int year);
  };

  // Accessors
  inline int Date::getMonth() const
  {
    return month;
  }

  inline int Date::getDay() const
  {
    return day;
  }

  inline int Date::getYear() const
  {
    return year;
  }
}

#endif /* DATE_H_ */

/*
 * Date.cpp
 *
 *  Created on: Oct 16, 2013
```

```
 *      Author: Nathaniel Gallinger
 */

#include <iostream>
#include <ctime>
#include "Date.h"
using std::cerr;
using std::cout;

// Default Constructor
NathanielGallinger::Date::Date()
{
  // Set date to current date
  const char monthIncrement = 1;
  const int yearAdd = 1900;
  time_t now = time(0);
  struct tm* tm = localtime(&now);
  month = tm->tm_mon + monthIncrement;
  day = tm->tm_mday;
  year = yearAdd + tm->tm_year;
}

// Constructor
NathanielGallinger::Date::Date(int month, int day, int year)
{
  if(checkDate(month, day, year)) {
    this->month = month;
    this->day = day;
    this->year = year;
  }
}

// Display function
void
NathanielGallinger::Date::display()
{
  cout << month << "/" << day << "/" << year << "\n";
}

// Check for valid date
int
NathanielGallinger::Date::checkDate(int month, int day, int year)
{
  const char VALID = 1;
  const char INVALID = 0;
  const char MAX_MONTH = 12;
  const char MIN_MONTH = 1;
  const char MIN_YEAR = 0;
  const char JAN = 1;
  const char FEB = 2;
  const char MAR = 3;
  const char APR = 4;
  const char MAY = 5;
  const char JUN = 6;
  const char JUL = 7;
  const char AUG = 8;
  const char SEP = 9;
  const char OCT = 10;
  const char NOV = 11;
  const char DEC = 12;
```

```
  const char MIN_DAY = 1;
  const char MAX_DAY_31 = 31;
  const char MAX_DAY_30 = 30;
  const char MAX_DAY_FEB = 28;

  int retval = VALID;

  // Verify month
  if ((month < MIN_MONTH) || (month > MAX_MONTH)) {
    cerr << "Month " << month << " invalid, must be in the range [1-12]\n";
    retval = INVALID;
  }

  // Verify year
  if (year < MIN_YEAR) {
    cerr << "Year " << year << " not valid, must be greater than or equal to zero\n";
    retval = INVALID;
  }

  // Verify day
  switch (month) {
  // fall through cases for days with 31 days
  case JAN:
  case MAR:
  case MAY:
  case JUL:
  case AUG:
  case OCT:
  case DEC:
    if ((day < MIN_DAY) || (day > MAX_DAY_31)) {
      cerr << "Day " << day << " not valid, must be in the range [1-31]\n";
      retval = INVALID;
    }
    break;
  // fall through cases for days with 30 days
  case APR:
  case JUN:
  case SEP:
  case NOV:
    if ((day < MIN_DAY) || (day > MAX_DAY_30)) {
      cerr << "Day " << day << " not valid, must be in the range [1-30]\n";
      retval = INVALID;
    }
    break;
  // check feb for 28, assume no leap year
  case FEB:
    if ((day < MIN_DAY) || (day > MAX_DAY_FEB)) {
      cerr << "Day " << day << " not valid, must be in the range [1-28]\n";
      retval = INVALID;
    }
    break;
  default:
    //  Invalid month
    retval = INVALID;
    break;
  }

  return retval;
}
```

```
/*
 * hw2.cpp
 *
 *  Created on: Oct 16, 2013
 *      Author: Nathaniel Gallinger
 */

#include "Date.h"
#include <iostream>
using std::cout;
using NathanielGallinger::Date;

int main()
{
  // Create date object and specify numbers
  cout << "Constructor with arguments 12, 12, 12: \n";
  Date date1(12, 12, 12);
  // Display object
  date1.display();

  // Test Accessors
  cout << "Accessor functions returning first object: \n";
  cout << date1.getMonth() << "/" << date1.getDay() << "/" << date1.getYear() << "\n";

  // Create date object with default constructor
  cout << "Default constructor returning today's date: \n";
  Date date2;
  date2.display();

  // Test error cases
  cout << "Attempt to call constructor with invalid arguments: \n";
  cout << "Case 1: 13, 2, 1900 \n";
  Date date3(13, 2, 1900);
  cout << "Case 2: 12, 34, 1900 \n";
  Date date4(12, 34, 1900);
  cout << "Case 3: 12, 15, -5 \n";
  Date date5(12, 15, -5);
}


Output:

Constructor with arguments 12, 12, 12:
12/12/12
Accessor functions returning first object:
12/12/12
Default constructor returning today's date:
10/17/2013
Attempt to call constructor with invalid arguments:
Case 1: 13, 2, 1900
Case 2: 12, 34, 1900
Case 3: 12, 15, -5
Month 13 invalid, must be in the range [1-12]
Day 34 not valid, must be in the range [1-31]
Year -5 not valid, must be greater than or equal to zero
```