SYSC 2002D Winter 2010 - Midterm (February 10th, 2010)
2:35pm-3:50pm (75min)
Closed book. No aids permitted other than what is provided.

The second last page of the exam contains extra space that may be used for any question.
You may remove and keep the last page of the exam.

Sample Solutions

Important Note: Comments are required on all questions for full marks!

Question 1 [10 marks]
Here is the bus database information similar to that of Chapter 1 in your notes. Note that the "`date`" struct in this question is unrelated to the Date class used in the other two questions on this test.

```
struct date {
    int day;  // 1 to 31
    int month;  // 1 = January, etc.
    int year; // in full (e.g. 2001).
 };

const int MAXOVERHAULS = 100;
struct busInfo {
    int number;  // fleet running number
    int model;
    date acquisitionDate;
    int numberOfOverhauls;
    date overhaulDates [MAXOVERHAULS];
};

const int MAXBUSSES = 1000;
struct busDB {
    busInfo info [MAXBUSSES];
    int actualBusses;
};
```

Given the above, write a function, `noWinterOverhauls`, that has one argument and returns an integer. The argument is the bus database (a `busDB` structure).

Your function is to return the number of busses that have had **no** overhauls in the winter. For purposes of this question, winter is December, January, and/or February. In other words, it returns the number of busses that have had either no overhauls at all, or all the overhauls were in March through November, inclusive.

```
int noWinterOverhauls(const busDB &db) { // 1 mark for signature
         // deduct ½ for missing const; and ½ for missing &

   int count = db.actualBusses; // assume no busses had a winter overhaul
                                // 1 mark for declaring / initializing count

   int curMonth; // temporary variable for month of current overhaul
                 // optional but makes life a lot easier

   // loop once for each bus (1 mark)
   for (int i=0;i<db.actualBusses;i++) {

      // loop once for each overhaul of the current bus (1 mark)
      for (int j=0;j<db.info[i].numberOfOverhauls;j++) {

         // store month of current overhaul
         curMonth = db.info[i].overhaulDates[j].month;

         // if month is December (12), January (1), or February (2)
         // then subtract this bus from the count as it has had a
         // winter overhaul and go on to the next bus (don't
         // check the remaining overhaul dates for this bus)
         // (5 marks for if, adjusting count, and break)
         if (curMonth==12 || curMonth==1 || curMonth==2) {
            count--;
            break; // required
         } // end if

      } // end inner loop (overhaul dates)
   } // end outer loop (busses)

   return count; // return the count (1 mark)
} // end function

// Notes:
// Deduct 1 mark if no comments
// Also see alternate solution on next page
```

```
// Alternate solution (same marking scheme as above)
int noWinterOverhauls(const busDB &db) {

   int count = 0; // assume all busses had a winter overhaul
   int curMonth; // temporary variable for month of current overhaul
   bool goodBus;

   // loop once for each bus
   for (int i=0;i<db.actualBusses;i++) {
      goodBus=true; // assume we should count this bus

      // loop once for each overhaul of the current bus
      for (int j=0;j<db.info[i].numberOfOverhauls;j++) {

         // store month of current overhaul
         curMonth = db.info[i].overhaulDates[j].month;

         // if month is December (12), January (1), or February (2)
         // then subtract this bus from the count as it has had a
         // winter overhaul and go on to the next bus (don't
         // check the remaining overhaul dates for this bus)
         if (curMonth==12 || curMonth==1 || curMonth==2) {
            goodBus=false;
            break; // optional but recommended
         } // end if
      } // end inner loop (overhaul dates)

      // if the bus had no winter overhauls then count it
      if (goodBus) count++;

   } // end outer loop (busses)

   return count;
} // end function
```

Question 2 [20 marks]
In this question, you are writing a program that **uses** the `Date` class. See the last page for the `Date.h` file.

You can assume the following program skeleton, and you need **only** write the code that goes where indicated:

```
#include "stdstuff.h"
#include "Date.h"

int main () {
  // your code goes here
  pause ();
  return 0;
}
```

Your program is to do the following:

1. output "Please enter today's date (DD MM YYYY): " and read in the date entered (Note that the user can enter any date here. You don't have to check that it's actually today.)
2. output "Today's date is: DD/MM/YYYY" where the date entered is displayed
3. output "Please enter a date (DD MM YYYY; today to exit): " and read in the date entered
4. if the user enters the same date as entered in step 1, output "Number of dates entered: " and the number of dates entered (**excluding** today's date entered in step 1 and here), and then terminate
5. otherwise:
    - output "You entered: " and the date entered
    - if the date entered is the same day of the week as the date entered as today's date, output "That date is the same day of the week as today.", otherwise output "That date is not the same day of the week as today."
    - if the date entered has the same month as today's date, output "That date has the same month as today.", otherwise output "That date does not have the same month as today."
    - output "That date is x days after today." where "x" is calculated appropriately (note that x may be positive or negative)
    - output "The date x days before today is: " and calculate and output that date ("x" is the same as in the previous step)
    - go back to step 3

You may assume that all dates entered are reasonable, and that dates are never moved outside the valid range (i.e. no error checking of dates is required).

```
Date today, current, daysBefore; // (1 mark)
    // today's date, current date we are dealing with, and date to move

int td, tm, ty, tdiw, d, m, y, diw, diff, numDates=0; // (1 mark)
    // day, month, year, day in week of today and current date;
    // difference; and number of dates considered so far

cout << "Please enter today's date (DD MM YYYY): ";
today.read(cin); // (1 mark for above 2 lines)

cout << "Today's date is: ";
today.write(cout); // (1 mark for above 2 lines)
cout << endl; // end the line (optional everywhere in program)

// get the day in week and day, month, year of today
tdiw = today.dayInWeek(); // (½ mark)
today.getDDMMYYYY(td,tm,ty); // (½ mark)
```

```
for(;;) { // loop (1 mark)
   cout << "Please enter a date (DD MM YYYY; today to exit): ";
   current.read(cin); // (1 mark for above 2 lines)

   // if the date is today, exit loop
   // can also use daysAfter or getDDMMYYYY and compare all 3 fields
   if(today.compareTo(current)==0) break; // (2 marks)

   numDates++; // increment number of dates considered (½ mark)

   cout << "You entered: ";
   current.write(cout); // (1 mark for above 2 lines)
   cout << endl; // end the line

   // get the day in week of current (½ mark)
   diw = current.dayInWeek();

   // compare days of the week (2 marks)
   if(tdiw==diw)
      cout << "That date is the same day of the week as today.\n";
   else
      cout << "That date is not the same day of the week as today.\n";

   // get the day, month, and year of current date (½ mark)
   current.getDDMMYYYY(d,m,y);

   // compare months (2 marks)
   if(tm==m)
      cout << "That date has the same month as today.\n";
   else
      cout << "That date does not have the same month as today.\n";

   // calculate days after (2 marks)
   diff = current.daysAfter(today);
   cout << "That date is " << diff << " days after today.\n";

   // find the date that many days before (2 marks)
   daysBefore = today; // so we don't change today
   daysBefore.move(-diff); // move temp date -diff days
   cout << "The date " << diff << " days before today is: ";
   daysBefore.write(cout);
   cout << endl;
} // end for

// output number of dates entered (½ mark)
cout << "Number of dates entered: " << numDates << endl;

// Note: Deduct 2 marks from total if there are no comments, and 1
// if there are few comments.
```

Question 3 [10 marks]
In this question, you are to add a new method to the Date class for which the header file is given on the last page. This method is intended for use by application programs. It is called `writeFull`. This method has no arguments and returns nothing (void). It is to output to cout the date it's applied to in the format shown.

Today would be written:
Wednesday, February 10th, 2010
The first day of January in 1900 would be written:
Monday, January 1st, 1900
etc.
In summary the format is: day of the week (starting with a capital letter), then a comma and space, then the month (starting with a capital letter), then a space, then the day of the month followed by "st", "nd", "rd", or "th" as appropriate, then a comma and space, then the year, then a new line.

Your code should be reasonably concise. As this method is part of the Date class, the implementation may use other methods in the Date class.

Additions to the Date.h file: **(2 marks)**

```
// writes the date to cout in format:
// Dayofweek, Month day-st/nd/rd/th, year <newline>
// e.g.: Wednesday, January 10th, 2010
// 1 mark for description
void writeFull() const; // 1 mark for signature (deduct ½ if no const)
```

Do the Date.h additions go in the private section or in the public section? **(1 mark)**

public (1 mark)

(Continued on next page.)

Additions to Date.cpp: **(7 marks)**

```
// writes the date to cout in format:
// Dayofweek, Month day-st/nd/rd/th, year <newline>
// e.g.: Wednesday, January 10th, 2010
void Date::writeFull() const { // 1 mark for Date::

   int d, m, y; // day, month and year of current date

   // days of the week (1 mark)
   // Note: could also have array of length 7 and subtract 1 from index
   String2002 days[8] = {"unused", "Monday", "Tuesday", "Wednesday",
                         "Thursday", "Friday", "Saturday", "Sunday"};

   // months of the year (1 mark) [or array of length 12]
   String2002 months[13] = {"unused", "January", "February", "March",
                         "April", "May", "June", "July", "August",
                         "September", "October", "November", "December"};

   // date endings (1st, 2nd, 3rd, etc.) (1 mark) [or array of length 31]
   String2002 ending[32] = {"unused", "st", "nd", "rd", "th", "th", "th",
          "th", "th", "th", "th", "th", "th", "th", "th", "th", "th", "th",
          "th", "th", "th", "st", "nd", "rd", "th", "th", "th", "th", "th",
          "th", "th", "st"}; // optional -- see below, too

   getDDMMYYYY(d,m,y); // break down date into day, month, year (1 mark)
   // note that students could instead use private function
   // dayNumberToDDMMYYYY(dayNumber,d,m,y) if they remember it

   // output date in the appropriate format (2 marks)
   cout << days[dayInWeek()] << ", " << months[m] << " " << d
        << ending[d] << ", " << y << endl;
   // deduct ½ mark for each missing comma, space or end line (endl, "\n")

   // instead of the ending array, could have something like:
   // if (d==1 || d==21 || d=31)
   //     cout << "st";
   // else if (d==2 || d==22)
   //     cout << "nd";
   // else if (d==3 || d==23)
   //     cout << "rd";
   // else cout << "th";

}

// Notes:
// Deduct 1 mark from total if no comments in implementation
// Deduct 1 mark from total if a 7 way statement is used for day of week
// Deduct 1 mark from total if a 12 way statement is used for month
// Deduct 1 mark from total if a 31 way statement is used for day ending
```

File Date.h: **You may remove and keep this page of the exam.**

```
class Date {

private:

  long dayNumber;  // 1 = Jan 1, 1900 and so on


public:

  // This class ensures that dates are always valid and always lie between
  // Jan 1, 1900 and Dec 31, 2099.
  // Attempts to move a date out of this range produce the appropriate limiting value.
  // Invalid constructor arguments, read errors, and so on result in Jan 1, 1900.

  // constructs a date containing Jan 1, 1900
  Date ();

  // constructs a date containing the specified date.
  Date (int day, int month, int year);

  // reads a date (dd mm yyyy format) from the given input stream. if an error of any sort
  // (bad read, invalid date) occurs, the input stream is left in the failed state.
  void read (istream &is);

  // writes a date to the specified output stream (in dd/mm/yyyy format)
  void write (ostream &os) const;

  // moves a date the specified number of days forward (positive day
  // values) or backwards (negative day values).
  void move (int days);

  // returns the day of the week (1 = Monday, etc.)
  int dayInWeek () const;

  // returns a date in dd, mm, yyyy form
  void getDDMMYYYY (int &day, int &month, int &year) const;

  // returns the difference between two dates.  this will be positive if the other
  // date is earlier than the implicit date (and negative if the other date is later)
  int daysAfter (const Date &otherDate) const;

  // returns 1 if the date is greater than (later than) the other date.
  // returns -1 if the date is less than (earlier than) the other date.
  // returns 0 if the two dates are equal
  int compareTo (const Date &otherDate) const;

};
```