SYSC 2002D Winter 2011 - Midterm (March 2nd, 2011)
11:35am-12:50pm (75min)
Closed book. No aids permitted other than what is provided.

Sample Solutions

Important Notes:

- **Comments are required on all questions for full marks!**
- **The second last page of the exam contains extra space that may be used for any question. If you use it, clearly indicate that next to the question you answer on the extra page.**
- **If you use the back of any pages for work that you want marked, indicate this clearly on both the back of the page that you used, and the front of the page opposite it.**
- **You may remove and keep the last page of the exam.**

Question 1 [10 marks]

Here is the student database information from Chapter 3 of your notes.

```
struct studentData {
      int studentNo;
      int yearsCompleted;
      int programCode;
      double GPA[4];
};

const int MAXSTUDENTS = 1000;
studentData info[MAXSTUDENTS];
int actualStudents;
```

Given the above database (that has already been set up for us), write a function, `statsGPA`, that has three parameters, and returns a double.

The parameters are:

1. the student database (an array of `studentData`)
2. the number of actual students
3. the highest GPA of any student in any year ("returned" by the function)

As well as "returning" the highest GPA of any student in any year, your function is to return the average GPA for all years by all students. For example, if we had 3 students, and the first two had completed 2 years, and the 3rd just one year, with GPAs of 10.0 and 11.0 (first student); 5.0 and 4.5 (second student); and 9.5 (third student); then the highest GPA would be 11.0, and the average would be (10+11+5+4.5+9.5)/5 = 8.0.

```
double statsGPA (const studentData db[], int num, double &maxGPA) {
   // 2 marks for signature: deduct ½ mark for missing const, &, []

   // declare count and sum (maxGPA declared above): 1 mark
   int count = 0; // deduct ½ if not int
   double sum = 0; // deduct ½ if not double

   // nested loop - outer loop for each student
   // inner loop for each GPA
   for (int i=0; i<num; i++) { // 1 mark
      for (int j=0; j<db[i].yearsCompleted; j++) { // 1.5 marks

         count++; // increment count: ½ mark
         sum += db[i].GPA[j]; // add current GPA to sum: 1 mark

         // if it's the first time or this GPA is the largest so far
         // update the maxGPA: 2 marks
         if (count==1 || maxGPA<db[i].GPA[j])
            maxGPA = db[i].GPA[j];

      }
   }
   return sum/count; // return the average: 1 mark
}

// Notes:
// Optionally could set maxGPA to 0 or negative at the start and then
// don't need to check for the first time.
```

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 your birth date (DD MM YYYY): " and read in the date entered. (It can be any date – you don't need to check that it's a reasonable birth date.)
2. output "Your birth 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 today's date (i.e. 2 3 2011), output "Number of dates entered: " and the number of dates entered (**excluding** the date entered in step 1 and the termination date), and then terminate
5. otherwise:
   - output "You entered: " and the date entered
   - if the date entered is before the birth date, output "That date is before you were born.", otherwise output "That date is on or after your birth date."

- if the date entered has the same day and month as the birth date (the year does not matter), output "That date is your birthday.", otherwise output "That date is not your birthday."
- output "On that date you are x days old." where "x" is calculated appropriately (note that x may be positive or negative – it should be negative if the date is before the birth date)
- if the date entered is the same day of the week as the birth date, output "The date is the same day of the week as your birth date", otherwise output "The date is not the same day of the week as your birth date"
- 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(2,3,2011), birth, current; // (1 mark)
    // today's date, birth and current date

int bd, bm, by, bdiw, d, m, y, diw, diff, numDates=0; // (1 mark)
    // day, month, year, day in week of birthdate and current date;
    // difference (optional); and number of dates considered so far

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

cout << "Your birth date is: ";
birth.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 birth date
bdiw = birth.dayInWeek(); // (½ mark)
birth.getDDMMYYYY(bd,bm,by); // (½ mark)

for(;;) { // loop (1 mark)
   cout << "Please enter a date (DD MM YYYY; 2 3 2011 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

   // see if date is before birth date (2 marks)
   if(current.compareTo(birth)==-1)
      cout << "That date is before you were born.\n";
   else
      cout << "That date is on or after your birth date.\n";

   // get the day, month, and year of current date (1 mark)
   current.getDDMMYYYY(d,m,y);
```

```
   // compare days and months (1 mark)
   if( bd==d && bm==m )
      cout << "That date is your birthday.\n";
   else
      cout << "That date is not your birthday.\n";

   // calculate days after (2 marks)
   diff = current.daysAfter(birth);
   cout << "On that date you are " << diff << " days old.\n";
   // note: diff variable is not needed

   // 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 your "
           << "birth date.\n";
   else
      cout << "That date is not the same day of the week as your "
           << "birth date.\n";

} // end for

// output number of dates entered (1 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.

The "+" method works with two Date objects and returns a Date object. The Date object returned has the highest day of the two dates, the highest month of the two months, and the highest year of the two years. If the resulting date is invalid, the method quits with an appropriate message.

Note that in writing this method, you can only use the methods on the last page. You may not use any of the private helper functions in the Date class.

Here is an example of a main program that uses Date's "+" method to help you understand how it works:

```
Date d1, d2, d3, d4, d5, d6;
// code to read in d1 and d2 goes here – assume d1 is 15 2 2000 and d2 is
// 12 4 2010
d3 = d1+d2;
// d3 now contains 15 4 2010 (as 15 is the largest day, 4 the largest month,
// and 2010 the largest year)

// code to read in d4 and d5 goes here – assume d4 is 31 1 1955 and d5 is
// 5 2 2011
d6 = d4+d5;
// this would cause the program to quit as 31 2 2011 is not a valid date
```

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

```
// Adding two dates returns a new date with the highest day, highest
// month, and highest year of the two dates.  If the new date is invalid,
// the method quits.  Neither date is changed by this method.
Date operator+ (const Date &otherDate) const;

// Notes: deduct ½ for each missing const and &;
// deduct ½ if quit not mentioned in comments;
// deduct 1 mark if no comments; deduct ½ mark if method name wrong
```

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

public

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

```
// Adding two dates returns a new date with the highest day, highest
// month, and highest year of the two dates.  If the new date is invalid,
// the method quits.  Neither date is changed by this method.
Date operator+ (const Date &otherDate) const {

   // day, month, and year for date, otherDate, and the maximums
   int d, m, y, od, om, oy, md, mm, my;

   Date default; // 1 1 1900

   getDDMMYYYY(d,m,y); // get day, month, year for date
   // this->getDDMMYYYY is ok, as is Date::getDDMMYYYY: 1 mark

   otherDate.getDDMMYYYY(od,om,oy); // get day, month, year for otherDate
                                    // 1 mark

   // find the maximum day, month, and year: 2 marks
   if (d>od)
      md = d;
   else
      md = od;
   if (m>om)
      mm = m;
   else
      mm = om;
   if (y>oy)
      my = y;
   else
      my = oy;

   // create the maximum date; note that this will be 1 1 1900 only if
   // both date and otherDate were 1 1 1900, or if we have an invalid date
   Date result(md,mm,my); // 1 mark
```

```
   // if result is 1 1 1900 and the original dates are not, we quit
   // 1 mark
   if ( result.compareTo(default)==0 &&
      ( compareTo(default)!=0 || otherDate.compareTo(default)!=0 ) ) {
      quit("Date::+ -- invalid date formed.\n");
   }

   // note that another way is that if the result is 1 1 1900 and the
   // original two dates are *not* the same (then both cannot be 1 1 1900)
   // we need to quit
   // if ( result.compareTo(default)==0 && compareTo(otherDate)!=0 ) {
   //   quit("Date::+ -- invalid date formed.\n");
   //}

   // if we get to here, all is ok, so return the result
   return result; // 1 mark

}

// Note: deduct 1 mark if no comments
```

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;

};
```