Carleton UNIVERSITY

Final
EXAMINATION
WINTER 2011

DURATION: 3 HOURS No. Of Students: **525**

Department Name & Course Number: **ECOR 1606 B, C, D Problem Solving and Computers**

Course Instructor (s): **Lynn Marshall, John Bryant, Andrew Marble**

AUTHORIZED MEMORANDA **Calculator**

Students MUST count the number of pages in this examination question paper before beginning to write, and report any discrepancy to a proctor. This question paper has **5** pages + cover page = **6** pages in all. The answer booklet provided has **8** pages.

This examination question paper may be taken from the examination room.

In addition to this question paper, students require: an examination booklet no
Scantron Sheet no

READ THE FOLLOWING INSTRUCTIONS BEFORE STARTING:

1. **Answer all questions in the answer booklet provided.**
2. **The answer booklet has an extra page at the end. If you need extra space for any question make a CLEAR note and put your work on the extra page.**
3. **There are six questions. Be sure that you find all of them.**
4. **Do not ask a question unless you believe that you have found a mistake in the exam paper. Exam questions will not be explained, and no hints will be given. You are limited to one question (unless it turns out there was in fact a mistake in the exam paper).**
5. **#include statements are not required.**

Question 1 (10 Marks)

a) Given the following function:

```
int func (int &a, int b) {

  int c;

  a = 7; b = 9; c = 10;
  return c * 5;
}
```

What would be output if the following code were executed?

```
int a = 12, b = 4, c = 6;

a = func (b, c);
cout << a << " " << b << " " << c << endl;
```

b) What would be output if the following code were executed?

```
int a = 8, b = 7, c = 10, d = 15;

if ((a < 6) || (c != 9)) {
  b++;
}
if ((d == 0) && (a != 8)) {
  b += 2;
}

cout << b << " " << d / 2 << " " << d % 4 << endl;
```

c) What would be output if the following code were executed?

```
int i, j;

for (i = 1; i <= 3; i++) {
  for (j = 1; j <= 3; j++) {
    if (j <= i) {
      cout << i << "," << j << ":";
    }
  }
}
cout << endl;
```

d) Show exactly what would appear if the following code were executed. Use # to represent blanks.

```
int a = 5; double b = 2.439;

cout << setiosflags (ios::fixed | ios::showpoint) << setprecision(2)
     << setw(4) << a << setw(8) << b << endl;
```

Question 2 (10 Marks)

a) After the execution of the statement below, what is the range of possible values for variable *x*?

```
double x;

x = 4 * rand() / (MAX_RAND + 1.0) + 3;
```

b) Write a statement that assigns a randomly chosen value between 2 and 8 (inclusive) to variable *y*. Each possible value should be equally likely.

```
int y;
```

c) Write code that adds up all of the values stored in array *data* and stores the result in variable *sum*. Be sure to declare any additional variables that you might require.

```
double data[20], sum;
```

d) Write an **expression** that evaluates to true if the value stored in variable z is evenly divisible by 5 and to false otherwise.

```
int z;
```

Question 3 (16 Marks)

Write a function that accepts an array of integer values and the number of values in this array. It should return true if any of the values in the array is the square of one of the **other** values, and false otherwise

example1:

```
the array contains { 4, 5, 9, -3, 25, 17 }
the function should return true  (25 = 5 squared, 9 = -3 squared)
```

example2:

```
the array contains { 4, 5, 12, -3, 24, 17, 1 }
the function should return false (none of the values are the square of one of the other values)
```

Note "**other** values". Having a "1" somewhere in the array does NOT guarantee a true result.

Your function must be consistent with the sample call below:

```
int values[20];

.....
if (checkForSquares (values, 20)) {
   .... // one of the values is the square of one of the other values
}
```

Question 4 (24 Marks)

File "data.txt" is supposed to contain between 10 and 100 integer values (inclusive of these limits) and all of the values are supposed to be different (i.e. there should not be any duplicate values). Write a program that reads this file and verifies that it is valid. Your program should terminate after having output exactly one of the following messages:

- "File cannot be opened"
- "File contains bad data"
- "File contains too many values"
- "File contains too few values"
- "File contains at least one duplicate value"
- "File is OK"

Question 5 (16 marks)

In many applications it is convenient to represent the days of a year using "day numbers". January 1st is day 1, January 2nd is day 2, and so on through December 31st which is either day 365 (if the year isn't a leap year) or day 366 (if it is).

Write a function that, given year and a day number, "returns" the corresponding month (January = 1, etc.) and day of the month (1 = first day of month, etc).

Example: If your function is given 2007 as the year and 32 as the day number it should "return" 2 as the month and 1 as the day of month (the day number corresponds to February 1st).

Assume that somebody else has provided you with a function that returns the number of days in a month. The prototype for this function is

```
int daysInMonth (int month, int year);  // month = 1 means January, etc.
```

You can just use this function in your function. You **DO NOT** have to write it.

Your function must be consistent with the following sample call:

```
cout << "Enter year and day number: ";
cin >> year >> dayNumber;
getMonthAndDay  (year, dayNumber, month, dayOfMonth);
cout << "That is day " << dayOfMonth << " of month " << month);
```

Question 6 (24 Marks)

Suppose an array containing the number of sick days taken by the employees of a company. The first array element contains the number of sick days taken by the first employee, the second element the number of sick days taken by the second employee. You are to write a function that takes this information and outputs a table like that shown below:

```
    Sick Days      Employees
----------------------------
     0 to  3          20
     4 to  7          11
     8 to 11           3
    12 to 15           6
    16 to 19           4
    20 or more         7
```

The sample table has six entries and the "bands" are 4 wide. This need not be the case.

Your function should accept

- the array containing the data
- the number of employees (i.e. the number of values in this array)
- the number of table entries required (maximum 12)
- the width of the "bands" to be used

It must be consistent with the sample call below.

```
// "sickDays" is the data array, "employees" is the number of employees
// in this case the table is to contain 6 entries and the bands are to be 4 wide
produceTable (sickDays, employees, 6, 4);
```