Carleton UNIVERSITY

ECOR 1606 A FINAL EXAMINATION
Fall 2007

DURATION: 3 HOURS **No. of Students: 74**

Department Name: Systems and Computer Engineering
Course Number: ECOR 1606 – Problem Solving & Computers
Course Instructor(s): Professor John Bryant

AUTHORIZED MEMORANDA
CLOSED BOOK, CALCULATORS NOT PERMITTED, DICTIONARY ALLOWED

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

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

In addition to this question paper, students require:	**an examination booklet**	**yes** ☒	**no** ☐
	a Scantron sheet	**yes** ☐	**no** ☒

READ THE FOLLOWING INSTRUCTIONS BEFORE STARTING:

1. There are four questions. Be sure that you find all of them. You may tear apart the question paper if you wish. Use the back sides of your examination booklet pages for rough work.

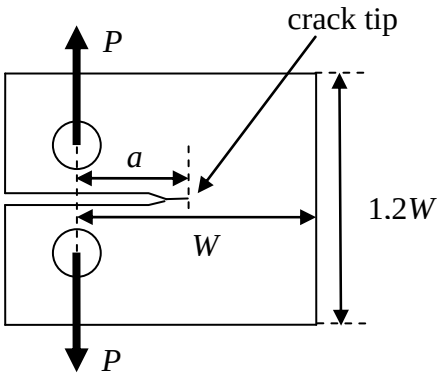
2. Do not ask a question unless you believe that you have found a mistake in the exam paper. Exam questions will not be explained, and no hints will be given. You are limited to one question (unless it turns out there was in fact a mistake in the exam paper). Once QA (question asked) has been written on the cover of you examination booklet no further questions will be answered.

3. You do not need to write any #include statements in any of the programs.

Question 1 (11 marks):

Compact tensile specimens like that illustrated below are used in investigating the propagation of cracks in engineering materials.

Depth of piece (into page) = *B*



Part I

The *stress intensity factor* (*K*) at the tip of the crack can be calculated using the formula below.

$$K = \frac{P}{B\sqrt{W}}\left[\frac{2 + a/w}{(1 - a/W)^{3/2}}\right]\left[0.866 + 4.64a/W - 13.32(a/W)^2 + 14.72(a/W)^3 - 5.6(a/W)^4\right]$$

where *K* is the stress intensity factor (in Pascal metres$^{1/2}$)
P is the applied force (in Newtons)
W and *B* are specimen dimensions (in metres)
a is the crack length (in metres)

Write a function that, given *P*, *B*, *W* and *a*, computes and returns *K*.

Part II

The specimen will fracture when *K* reaches the specimen's *fracture toughness* (K_{1C}, same units as *K*). Write a function that, given *P*, *B*, *W* and K_{1C}, computes and returns the critical crack length (the crack length at which *K* is equal to K_{1C}). Your function must be efficient and the value returned must be within 0.01mm of the correct value.

Question 2 (11 marks)

Part I

Write a function that, given an array of integer values and the size of this array, returns the position in the array of the lowest value. If the lowest value occurs two or more times, the position of the first occurrence should be returned (see the third example).

Examples:	array = { 5, 7, 3, 12, 1, 9, 15 }	function should return 4
	array = { -1, -9, 3, 6, 34, 0 }	function should return 1
	array = { 12, 45, 67, 23 12, 96 }	function should return 0

Part II

A park has five picnic areas (known as area 0 through area 4). Each has room for 100 cars. Normally cars arriving at the park simply head for one of these areas, with each of the possibilities being equally likely. This can result in one area overflowing while there is still space at others. To get around this problem, the park service has put up an electronic sign, where the roads to the five areas diverge, indicating how many cars have gone to each of the five areas. Unfortunately this sign is somewhat obscured and only 34.5% of cars notice it. Cars that do notice the sign will head for the area with the least number of cars if the area that they would otherwise have gone to is quite full (90 or more cars).

Write a program that computes the probability that one or more of the picnic areas will overflow on a day when 480 cars arrive. To facilitate marking (and to give you a chance to demonstrate your familiarity with functions) your program must consist of

1/. A function that simulates the arrival of 480 cars and returns true if any one of the picnic areas overflows.

3/. A main function that determines and outputs the required probability (expressed as a percentage).

Notes:

If two areas are equally empty, it really doesn't matter which one is chosen. You can assume that, in such cases, the lowest numbered area will always be picked.

An area can have 90 or more cars and still have the least number of cars.

Your part II solution should make use of your part I function.

Question 3 (11 marks)

Suppose that we have a file containing daily closing values for the Toronto Stock Exchange (TSX) composite index. The short sample below should give the idea. On the first day the market closed at 14523.890, on the second day it closed at 14378.345, and so on.

```
14523.890
14378.345
14405.491
14413.629
….
```

We are interested in the market's volatility (how much closing values vary from day to day). Write a program that reads in the file and produces a table like that shown below.

```
 Absolute Day to Day Change   Percentage of Days
-------------------------------------------------
   0.0 <= change < 0.2%             22.52
   0.2 <= change < 0.4%             16.56
   0.4 <= change < 0.6%              9.45
   0.6 <= change < 0.8%             10.56
   0.8 <= change < 1.0%              8.45
   1.0 <= change < 1.2%              8.34
   1.2 <= change < 1.4%              9.21
   1.4 <= change < 1.6%              6.45
   1.6 <= change < 1.8%              4.23
   1.8 <= change < 0.2%              3.23
   change >= 2.0%                    1.00
```

The absolute percentage change from one day to the next is defined as

$$\left| \frac{\text{second day closing value - first day closing value}}{\text{first day closing value}} \right| \times 100$$

For the sample data file, the absolute percentage change from the first day to the second day is 1.002%.

Have your program read in the name of the file to be processed. If an input error is detected, your program should just output an error message and terminate. The file does not contain a special "end of data" value. Instead your program should keep on reading data until the end of the file is encountered. The format of the table produced must match that of the sample table (in a general way – don't waste time counting columns) and all values must be output with the same number of decimal places.

Note: While reading all of the values in the file into an array is one way of approaching this problem, this is NOT the best approach (because the size of the array imposes a limit on the number of values that can be processed). There is a better way. Students who read all of the values in the file into an array will lose 1 mark.

Question 4 (12 marks)

Researchers performing an experiment have obtained a number of data points. Each data point (the see diagram below) involves an x co-ordinate, a y coordinate, and a textual tag.

The researchers have begun writing a data processing program and have got as far as defining the following structure:

```
struct dataPoint {
   double x, y;
   char tag [10]; // tags can be up to 9 characters long
};
```

They need you to help out by writing three functions.

Part I:

Tags consist of a researcher id, a hyphen, and a reading number. Write a function that, given a tag, copies the researcher id contained within it into another string variable. The following sample call should give the idea.

```
extractId (tag, id);  // copies the researcher id portion of "tag" into "id"
```

Part II:

Write a function that, given two points, returns the distance between them. You are expected to either know the formula or be capable of working it out (think Pythagoras).

Part III

Write a function that, given an array of data points and the length of this array, determines and returns the distance between the two points **obtained by different researchers** that are closest to each other. Your function must also "return" the positions (within the array) of these two points. The following sample call may help make things clear.

```
dataPoint array[200]; // array of points
int n; // actual number of points in array
int i, j;
double shortestDistance;

// code omitted (assume that values are somehow placed in the array)

shortestDistance = findShortestDistance (array, n, i, j);

cout << "The two closest points by different researchers are "
     << shortestDistance << " apart." << endl
     << "These points are stored at array positions " << i << "and " << j << endl;
```

You may (and should) make use of the functions of part I and part II.

If the array contains fewer than 2 data points or if all of the data points were obtained by the same researcher, your function should return -1 (and not bother with the array positions).