

## Exam 2

*Due: 2:30 PM May 8, 2008*

- **You must work alone on this exam. If you have any questions, ask the TAs or the professor, but do not discuss the exam at all, with anyone else.** Your work should be based on the material in the course and the course materials (including everything on the cs004 website and discussion board); do not look at other sources, including the Web.
- **Do not post to the discussion board while the exam is out.** Visit a TA on hours or mail questions to `cs004tas@cs.brown.edu`. Read the discussion board, however, as clarifications will be posted there.
- **To receive a grade above 0 on the exam, abide by the following statement. Then print out a copy of this page, sign the statement and turn it in to the cs004 hand-in bin (the one on the second floor by the stairs). We really won't give you a grade on the exam if this is not done.**

All work on this exam is my own. I have not discussed the problems or their solutions with anyone except the course staff. I did not consult any materials other than those officially provided by the course staff.

Login (please print): \_\_\_\_\_

Signed: X\_\_\_\_\_

Prob.	Poss. Points	Points	Grader
1	15		
2	20		
3	20		
4	25		
5	20		
Total	100		

## Tips

- For questions 4 and 5 we will provide stencil code for you via the course webpage. **You must use this code** as a starting point for your work. You should hand in files with exactly the same names as the ones we provide, and you should not change any of the provided support code.
- For each problem we have given you the name of the program to write, and in some cases we've specified the name of a function to include in your program. **Do not change the names of any functions or files from those given in the problem.** You are welcome to define and use any helper functions you want. You are also welcome to use anything we've defined or used in any previous lecture or assignment.
- Comment your code thoroughly and format your code appropriately by following the C Formatting and Coding Reference on the *Resources* section of the course website. You will be penalized for failing to do so.
- Simplify your code and make sure it's clean and elegant. As in homeworks, we are always looking for the best answer on exam problems.
- Don't wait until the last minute. Get started early, this way you can take advantage of TA hours throughout the week.

## Submitting your exam

To submit your code electronically, make sure that in your `home` directory you have a `course` directory and that in that there is a `cs004` directory. You should save your code in a directory in your `cs004` directory called `finalexam`. To summarize, you will place all your work into the directory

```
/u/<login>/course/cs004/finalexam
```

*Don't forget to change <login> to be your login name.* After you have your work saved in the proper directory you can run the hand-in script. Open a terminal, then type

```
cs004_handin finalexam
```

**CS004**

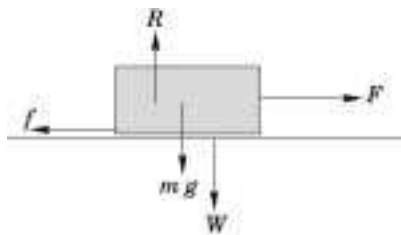
Exam 2

**2:30 PM May 8, 2008**

If you have any questions about submitting your work, email your question to [cs004tas@cs.brown.edu](mailto:cs004tas@cs.brown.edu) or ask a TA during hours. No late submissions will be accepted.

### Problem 1

Write a C program `friction.c` that computes the necessary force to move a solid block across a plane where friction resists the movement of the block as shown in the figure below.



Because friction is the only force acting on the block, the required force  $F$  to move the block (in Newtons) is equal to the frictional force  $f$  given by  $f = \mu mg$  where  $\mu$  is the coefficient of friction,  $m$  is the mass of the block, and  $g$  is the force due to gravity ( $9.8 \text{ m/s}^2$ ). In this problem  $\mu = 0.35$ . The mass of the block will be computed from the volume and the density. Your program should prompt the user to enter the height, length, and width of the block and then calculate the volume. The mass is given by

$$m = dV$$

where  $d$  is the density of the block ( $8,960 \text{ kg/m}^3$  for this program), and  $V$  is the volume of the block. After reading the dimensions of the block and calculating the force  $F$ , your program should print the dimensions and the force. A sample run of your program should look something like the following:

```
Enter the block height (m): 1
Enter the block length (m): 2
Enter the block width (m): 2
```

```
For a block with dimensions
  Height = 1.000000
  Length = 2.000000
  Width = 2.000000
the required force is 122931.203125N
```

**Problem 2**

Write a C program `arraysums.c` that reads in a  $5 \times 5$  array of integers and then prints out the row sums and the columns sums. A sample run of the program should look like this:

```
Enter row 1: 8 3 9 0 10
```

```
Enter row 2: 3 5 17 1 1
```

```
Enter row 3: 2 8 6 23 1
```

```
Enter row 4: 15 7 3 2 9
```

```
Enter row 5: 6 14 2 6 0
```

```
Row totals: 30 27 40 36 28
```

```
Column totals: 34 37 37 32 21
```

**Problem 3**

The square root  $x$  of a number  $a$  can be computed by successive approximations using the following iterative formula:

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right)$$

Start with  $x_1 = 1$  as an initial (very rough) approximation for the square root of  $a$ . Given an approximation  $x_n$ , a better approximation  $x_{n+1}$  can be computed using the above formula. Compute  $x_1 \dots x_n$  until the square of  $x_n$  is close enough to  $a$ . That is, until:

$$|x_n^2 - a| \leq e$$

where  $e$  is the desired precision of the approximation.  $x_n$  is an approximation of the true square root  $x$ .

Write a C program `sqroot.c` to approximate the square root of a number using the procedure described above. Prompt the user to enter the number  $a$  and the desired precision. For example, a typical execution of the program might look like:

```
Enter the number a: 24
Enter the precision: .0001
Square root of 24.000000 is approximately 4.898979
```

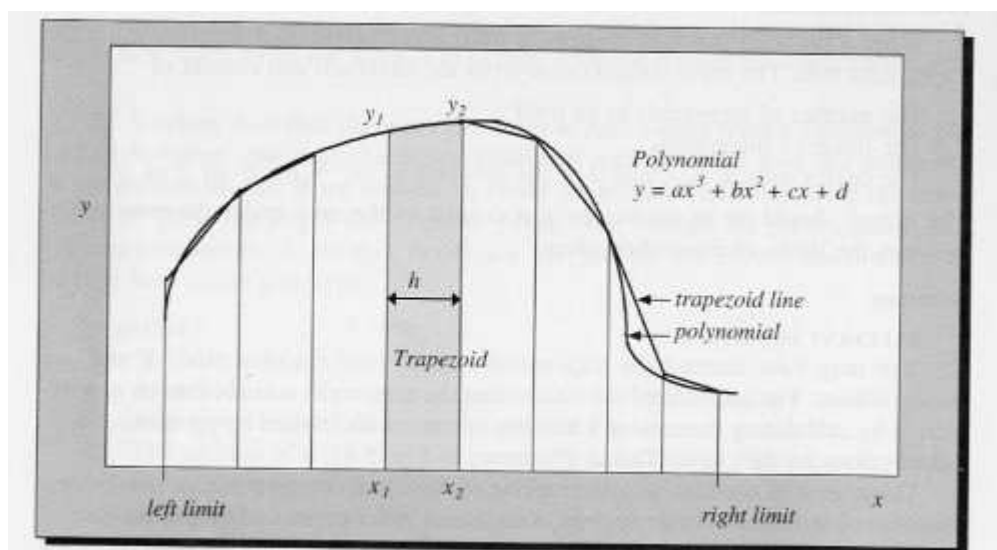
Use type `double` for the floating point variables. The absolute value can be found with the function `fabs()` in the math library. Be sure to insert `#include <math.h>` in your program and compile it with the `-lm` command as follows:

```
gcc -Wall -o sqroot sqroot.c -lm
```

### Problem 4

In this problem you will write a C program to integrate a third order polynomial  $ax^3 + bx^2 + cx + d$  using the trapezoidal rule. Your main function will ask the user for some input parameters and then call a function to do part of the work.

The area under a curve can be approximated by summing the areas of a number of trapezoids formed by connecting individual points on the curve as shown in the figure below.



In some regions of the curve, a trapezoid is a good representation of the area. In other regions, it is not. As more (and therefore narrower) trapezoids are used, the approximation improves. The area  $A$  of the shaded trapezoid shown in the figure is

$$A = h(y_1 + y_2)/2$$

If we want to use  $n$  trapezoids, the value of  $h$  is found by  $h = (r - l)/n$  where  $r$  is the right limit and  $l$  is the left limit of the integration. For each trapezoid we can calculate the corresponding  $y_1$  and  $y_2$  value from

$$\begin{aligned} y_1 &= ax_1^3 + bx_1^2 + cx_1 + d \\ y_2 &= ax_2^3 + bx_2^2 + cx_2 + d \end{aligned}$$

With  $h$ ,  $y_1$ , and  $y_2$  we can find the area of each trapezoid.

We are providing you some code to get you started. The stencil code can be found on the website, or you can copy it from the course directory with the command:

```
cp /course/cs004/pub/finalexam/trapezoid.c ~/course/cs004/finalexam/.
```

You will fill in the function `double trapezoid(double h, double x1, double x2, double a, double b, double c, double d)` that calculates the area of a single trapezoid with those parameters. Use variables of type `double` for all the floating point variables. Your main function should prompt the user to enter the number of trapezoids to use in the integration, the limits of the integration (left and right), and the four coefficients of the polynomial  $a$ ,  $b$ ,  $c$ , and  $d$ . It should then determine  $h$  and repeatedly call `trapezoid` to calculate the area of each trapezoid, keeping track of the total sum of all the trapezoids. The final sum is the approximation of the area under the curve. A sample run of your program that integrates the polynomial  $4.1x^3 - 2x^2 + 3.5x + 1$  on the interval from 0 to 1.5 with 5 trapezoids should look similar to this:

```
Enter n, left limit, right limit: 5 0 1.5
Enter a, b, c, d: 4.1 -2 3.5 1
Area = 8.539125
```

## Problem 5

In class we looked at how a linked list can be built with a C program to efficiently store and organize a large amount of data. In this problem you will write a very basic linked list program that allows a user to insert new nodes and print the list. Each node in the linked list will contain one integer datum.

We are providing you some code to get you started. The stencil code can be found on the website, or you can copy it from the course directory with the command:

```
cp /course/cs004/pub/finalexam/linkedlist.c ~/course/cs004/finalexam/.
```

We have written the `main()` function for you. It takes care of displaying a menu of options to the user and calling the appropriate functions to operate on the linked list. We've also written the typedefs for `Node` and `List`. It is your task to fill in the code for the two remaining functions. The first function, `List InsertAtEnd(List list, int value)` appends a node containing the integer `value` to the end of the linked list `list`. The second function, `void PrintList(List list)`, will print out all the integers in the list (in order, and separated by spaces).

We've also provided a demo version of this program to show you how it should run when you are finished filling in the two functions. You can run the demo with the following command:

```
/course/cs004/bin/cs4_linkedlist_demo
```

Remember to consider the following things:

- You must allocate memory for each new node you create.
- The `list` you pass into `InsertAtEnd` could be empty or non-empty.
- In `InsertAtEnd`, after appending a node to `list`, make sure you return the entire list.
- The `next` field of a node at the end of a linked list should point to `NULL`.