University of Toronto Scarborough

April 2015 Examinations

CSC B09H3S

Duration: 3 hours

Aids allowed: Any books and papers.
No electronic aids allowed: No calculators, cell phones, computers, ...
To pass the course you must receive at least 35% on this exam.

If you are writing this exam at the usual time and place, you have received a label with your seat number and name on it. Attach that label here, over this text. Please remove all of the backing.

Otherwise, please write your name and student number, underlining your surname:

Make sure you have all 11 pages (including this page).
(Don't panic about the page count—there's lots of space for answers.)

Answer *all* questions. Answer questions in the space provided. Answers not in the correct space will not be graded unless a note in the correct space says "see page ..." and the answer on that page is clearly labelled with the question number.

Be careful not to get stuck on some questions to the complete exclusion of others. The amount of marks or answer-space allotted does not indicate how long it will take you to complete the question, nor does the size of the answer-space indicate the size of the correct answer.

In general, in the C programming questions you can omit the #includes, and you can omit comments unless you need to explain something unclear about the functioning of your code.

Do not open this booklet until you are instructed to.

Do not write anything in the following table:

question	value	grade	question	value	grade
1	15		5	16	
2	8		6	15	
3	9		7	15	
4	16		8	6	
subtotal			total	100	

1. [15 marks]
All of the “accounts” on a unix/linux system are represented by a file named “/etc/passwd”.
Lines in /etc/passwd look like this:

```
ajr:x:300003:500:Alan Rosenthal:/cmshome/ajr:/bin/bash
```

The sixth field is the home directory.
The *cut* command takes a “−d” option which says what the field separator is, so the sixth field could be extracted with “cut −d: −f6”.

Write a shell script which outputs the percentage of the entries in /etc/passwd which have a home directory somewhere under /cmshome. If useful, you can assume that there are no spaces in any home directory names.

Possible hint: Remember the *case* statement, which can match “glob” patterns such as “/cmshome/*”.

Be sure to write backquotes and single-quotes correctly and distinctly. We can only grade what you wrote, not what you meant.

- A single-quote looks like this: ' or this: ´
- A backquote looks like this: `

2. [8 marks]
A programmer writes the statement

```
      test "$x" = "$y"
```

as an 'if' condition. Remember that '=' is a string comparison in *test*.

a) Why are the double quotes necessary? Explain by giving possible values for the variables *x* and *y* such that "test $x = $y" would yield a syntax error from *test*, but the above version with the double quotes would work as expected.

x is:

y is:

b) Using a version of test written along the lines of our "simpletest", the programmer gets a syntax error anyway, even with the double quotes! That's because the variable *x* contained the two character string "−f", so "=" was considered to be a file name, and the value in *y* was then a syntax error. How would you change the above *test* statement to protect against this possibility?

continued...

3. [9 marks]
State three distinct items of information (data) in the inode of a file in the unix filesystem (you needn't give their names from <sys/stat.h>; just describe them with a few words each).

a)

b)

c)

4. [16 marks]
Write a complete C program (except that you can omit the #includes) which takes one mandatory command-line argument, which it expects to be a directory. Other than "." and "..", that directory is expected to contain only plain files. Your program reads all of the files in that directory and outputs the total byte count.

You must do all usual error checking and output appropriate error messages.

Hint: If you chdir() to the directory, you can use the file names from readdir() as is.

(more space for your question 4 answer, if required)

continued...

5. [16 marks]
The *diff* program contains a sophisticated algorithm which decides which lines to compare to which. For a “trivial diff” program for this exam question, we evade this algorithm by assuming that files all have at most one line (n.b. *at most*; they could still be empty).

Write a complete C program (except that you can omit the #includes) which takes two filenames (no other possibilities, no options) and behaves as diff except that it reads only the first line (if available). You can assume that the first line is a maximum of 1000 chars long, although you may not exceed array bounds even if it isn’t. And all usual error checking and error messages are required.

If the files are identical, there is no output. Otherwise, if the first file is empty and the second consists of the line “foo”, *diff* will output:

```
0a1
> foo
```

If the first file consists of the line “foo” and the second file is empty, *diff* will output:

```
1d0
< foo
```

If both files consist of one line but it is different, *diff* will output:

```
1c1
< foo
---
> bar
```

You can start your program below, but there is additional room on the next page if needed.

continued...

(more space for your question 5 answer, if required)

continued...

6. [15 marks]
Write a complete C program (except that you can omit the #includes) which calls fork(); the child executes the command "tr e f <file1 >file2" (without using *sh* or system()); and the parent calls wait() and outputs the child's exit status in the exact format "child process exit status was %d". You may hard-code the fact that 'tr' is /usr/bin/tr. All usual error checking and error message reporting is required.

continued...

7. [15 marks]
Write a complete C program (except that you can omit the #includes) which listens on port 2000 and receives messages, as follows: Anyone can connect to port 2000 and send data, and that data is output on the standard output, without alteration. You do not need to worry about the network newline conversion.

Once a client connects, you read all data from it in a loop until end-of-file (caused by the client disconnecting), then you loop around for the next client; you do not need to handle multiple clients simultaneously.

continued...

8. [6 marks]
From a high-level view, main() is the first thing executed in a C program. But there has to be some code which calls main().

When your program is executed in unix, it starts executing at a certain address. Code known as "crt0" (for "C runtime, the very first thing") sets up various things needed by the C library, and calls main() with the appropriate arguments.

The last line of crt0.c (before a closing brace) is usually something very similar to:

```
      exit(main(argc, argv));
```

What is this line doing?

continued...

Extra space if needed
(you must write “see page 11” in the usual answer space for the given question)

End of exam. Total marks: 100. Total pages: 11.