On my left is:	**MY last name**	On my right is:
____________________ (last name/aisle/empty)	____________________________	____________________ (last name/aisle/empty)

CSC 352 Mid-term Exam
Monday, October 12, 2015

READ THIS FIRST

Read this page now but do not turn this page until you are told to do so. Fill in your last name in the MIDDLE box above, and get the names of your classmates, if any, on each side of you, too.

This is a 45-minute exam with a total of 100 points of regular questions and an extra credit section.

To avoid distractions for those who are still working, I ask that nobody leaves their seat during the last five minutes of the exam. If you finish during that last five minutes, please remain quietly seated—no "packing up", checking phones, etc.—until the exam is over.

You are allowed no reference materials whatsoever.

If you have a question, raise your hand. We will come to you. DO NOT leave your seat.

If you have a question that can be safely resolved with a minor assumption, such as the name of an option for a UNIX command, state the assumption and proceed.

Don't make problems hard by assuming that they need to do more than is specifically mentioned in the write-up, or by assuming that the solution that comes to mind is "too easy."

If you're stuck on a problem, please ask for a hint. Try to avoid leaving a problem completely blank—that's a sure zero.

It is better to put forth a solution that violates stated restrictions than to leave a problem blank—a solution with violations may still be worth partial credit.

When told to begin, **scribble your initials in the right hand corner of the top side of each sheet, checking to be sure you have all SIX sheets.** (Exams are scanned, to inhibit the temptation of altering graded work, and once in a while the sheets of two exams become intermixed when scanning.)

Remember that we'll have a sign-out process when then exam is over. If you have a tight schedule, make that known, and we'll get you to the front of a line.

Problem 1: (14 points)

For this problem you are to write a **Java program** named `argGrep`. Unlike `fgrep` and your own `mgrep`, which search files or standard input for a string, `argGrep` searches its second and following command-line arguments for occurrences of its first command-line argument. It prints arguments that match, one per line. Example:

```
% java argGrep in testing looks inward
testing
inward
% java argGrep out testing looks inward
% java argGrep 0 $(seq 30)
10
20
30
%
```

My solution uses `String`'s `boolean contains(String)` method. Here's an example of usage from `javarepl.com`:

```
java> "testing".contains("in")
java.lang.Boolean res0 = true

java> "testing".contains("out")
java.lang.Boolean res1 = false
```

`argGrep` assumes it is run with at least two arguments.

Here's some "boilerplate" to get you started with `argGrep`. **Note: you might not need all of it.**

```
import java.io.*;
public class argGrep
{
    public static void main(String args[]) throws IOException
    {
        BufferedReader in =
          new BufferedReader(new InputStreamReader(System.in));
```

Problem 2: (14 points)

Write a **C program** `firstx` that outputs the zero-based position of the first occurrence of an `'x'` (lower-case) on standard input. If there is no `'x'`, `firstx` prints `-1`.

```
% echo "Texas text test" | firstx
2
% echo "xyzzy" | firstx
0
% echo "testing" | firstx
-1
% cal 2015 | firstx
-1
```

Implementation note: You can terminate execution in C by calling `exit(0)`. A prototype for `exit` is in `stdlib.h`. Be sure to have `#includes` for both `stdio.h` and `stdlib.h`.

Problem 3: (9 points)

Write a **bash script** named `mvqt` that when given an argument such as `alan-questions.txt` it creates a directory named `alan`, and moves `alan-questions.txt` into `alan/questions.txt`.

Example:

```
% ls -l
-rw-r--r-- 1 whm staff 1103 Oct  9 19:47 alan-questions.txt
-rw-r--r-- 1 whm staff 1968 Oct  9 19:48 tim-questions.txt

% mvqt alan-questions.txt

% ls -l
drwxr-xr-x 3 whm staff  102 Oct  9 19:48 alan
-rw-r--r-- 1 whm staff 1968 Oct  9 19:48 tim-questions.txt

% ls -l alan
-rw-r--r-- 1 whm staff 1103 Oct  9 19:47 questions.txt
```

Recall that the first command-line argument for a script can be accessed as `$1`. My solution for this problem uses command substitution, `$(...)`, which you likely used in `getrc`, on assignment 3.

Note that the general form of `mvqt`'s argument is *`NETID`*`-questions.txt`. Assume that *`NETID`* does not contain a `'-'`; you won't see something like `smith-jones-questions.txt`. Assume that `mvqt` is run with one argument.

Problem 4: (3 points)

Imagine that you've got a directory with dozens of *`NETID`*`-questions.txt` files:

```
% ls
alan-questions.txt     greg913-questions.txt  sw-questions.txt
alex93-questions.txt   harris-questions.txt   tim-questions.txt
...
```

Write a bash `for`-loop that runs `mvqt` on each of the `*-questions.txt` files in the current directory in turn. (Note that "`mvqt *-questions.txt`" would process only the first argument.)

Problem 5: (14 points)

Write a **C program** named `oxo.c` that reads integers from standard input and for each integer outputs a square of the specified size with alternating `O`s and `X`s. Example:

```
% echo 5 3 | oxo
OXOXO
XOXOX
OXOXO
XOXOX
OXOXO

OXO
XOX
OXO

%
```

Following the example used in `rectangle.c` on `a6`, use the following `while` as your main loop. Note that it stores each integer read in an `int` named `size`, so have `int size;`.

```
while (scanf("%d", &size) == 1) { ... }
```

Any number of integers may be read but assume values are always an odd number greater than zero. Note that the first letter of each square is `O`, and that each square is followed by an empty line.

For one point of extra credit, don't use "`if`" in your solution.

I'll save you a moment on this one: don't bother with any `#include`s.

Problem 6: (9 points)

On assignment 3 you wrote `clp.java`, a program that analyzed command lines for arguments, redirection, piping, etc. On this problem you are to do that same analysis by hand.

Here is a completed example:

```
% ls -l -R | wc > x
```

Command 1: `ls` Arguments: `-l`, `-R` Stdin: `K` Stdout: `P`	Command 2: `wc` Arguments: `none` Stdin: `P` Stdout: `x`	Command 3: Arguments: Stdin: Stdout:

Start by filling in the command name(s). `ls` and `wc` are the two commands in the pipeline above.

Separate arguments with a comma; two lines are provided. Write "`none`" if no arguments.

Use "`K`" when Stdin (standard input) is the keyboard. Use "`S`" when Stdout (standard output) is the screen. Use "`P`" when Stdin or Stdout is a pipe. Hint: Circle each redirection operator and its operand, to keep from confusing them with arguments or the command.

Three boxes are provided for every command line but some lines have only one or two commands. Leave any extras blank.

Important: assume there are no errors/typos in the command lines!

Command line:

```
% ls x y | wc -c | cat -n
```

Command 1: Arguments: Stdin: Stdout:	Command 2: Arguments: Stdin: Stdout:	Command 3: Arguments: Stdin: Stdout:

(continued on next page)

(Problem 6, continued)

Command line:

```
% wc < a b > c d
```

Command 1:	Command 2:	Command 3:
Arguments:	Arguments:	Arguments:
Stdin:	Stdin:	Stdin:
Stdout:	Stdout:	Stdout:

Command line:

```
% java java java | wc wc -pipe ls > cat
```

Command 1:	Command 2:	Command 3:
Arguments:	Arguments:	Arguments:
Stdin:	Stdin:	Stdin:
Stdout:	Stdout:	Stdout:

Command line:

```
% < a b | e f-g h | > x y z
```

Command 1:	Command 2:	Command 3:
Arguments:	Arguments:	Arguments:
Stdin:	Stdin:	Stdin:
Stdout:	Stdout:	Stdout:

DOUBLE CHECK: Do you have an entry for all four fields for each command on each command line?

Problem 7: (12 points)

Each time the Tester is run it records when it was run, what was tested, and more. Here's a short sample of a simplified version of those entries:

```
% cat log
Wed Sep 23 20:11:06 MST 2015 args: patlen
Wed Sep 23 20:11:15 MST 2015 args: lpp
Wed Sep 23 20:35:58 MST 2015 args: patlen.java
Wed Sep 23 21:39:24 MST 2015 args: lpp
Wed Sep 23 22:18:34 MST 2015 args: lpp.java
Wed Sep 23 22:43:17 MST 2015 args: llp.java
Wed Sep 23 23:03:44 MST 2015 args: lpp
Wed Sep 23 23:15:38 MST 2015 args: patlen.java
Wed Sep 23 23:26:07 MST 2015 args: lpp.java
Wed Sep 23 23:58:23 MST 2015 args: lpp
```

Write a **one-pipeline bash script, `rbh`** (runs by hour), that reads lines like the above on standard input and outputs a count of how many runs were made in each hour of the day. For the file above, here's the output:

```
% rbh < log
      3 20
      1 21
      2 22
      4 23
```

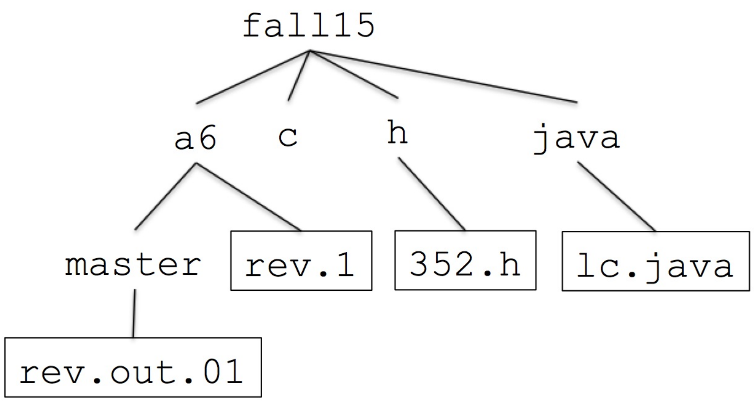
The first line shows that there were three runs between `20:00:00` and `20:59:59`. There was only one run between `21:00:00` and `21:59:59`, and so forth. `rbh` produces at most 24 lines of output.

My solution uses `cut`, but remember that `cut`'s `-d` option allows only one delimiting character to be specified.

My solution also uses `uniq -c`, which you may recall can be used to count occurrences in sorted input:

```
% cat colors
green
green
red
blue
green
red
% sort < colors | uniq -c
      1 blue
      3 green
      2 red
```

Problem 8: (8 points) (1 point each)



This problem is like the `tree-*` problems on assignment 3. Given the tree on the right, write a bash command line to answer each of the questions below.

Each question or problem starts with "In *`DIRECTORY`*, ..." meaning that you should assume that *`DIRECTORY`* is your current working directory.

<u>IMPORTANT:</u> You may use multiple commands to answer a question but **<u>you may NOT change your working directory</u>**. That is, an answer like "**`cd`** `../h; wc -l *.c`" will be considered wrong.

Consider the word "file" to mean all type of entries—regular files, directories, and more. For example, "`ls | wc -l`" is a correct answer for "How many files are in the current directory?" even if the current directory contains subdirectories. Also, don't worry about hidden entries.

A few individual files, surrounded with a box, are shown in the tree above but **<u>assume that each directory might have a number of files that are not shown</u>**.

(1) In `master`, how many lines are in `rev.1`?

(2) In `fall15`, what lines in `352.h` contain the string "`define`"?

(3) In `java`, what is the last line in `rev.out.01`?

(4) In `java`, how many `.pdf` files are in `fall15`?

(5) In `c`, how many files in `java` do NOT have a `'.'` in their name?

(6) In `fall15`, how many files in `c` have a name that both starts **<u>and</u>** ends with an `'a'`?

(7) In `master`, copy your `.bashrc` to the current directory <u>using as few characters as possible</u>.

(8) In `master`, is `cd ../../fall15` valid? (If not, correct the path to reach `fall15`.)

Problem 9: (5 points)

For one point each, answer the following questions, all related to C.

(1) In as few characters as possible, what is the value of `x++`?

(2) Write an `if` statement that's valid in C but not in Java.

(3) What is the value of the following C expression? `10 > (20 > 30)`

(4) What are the three aspects of a Java or C expression that `whm` says are usually the most useful to consider?

(5) On a 24-bit machine that has 12-bit bytes and uses one's complement arithmetic for computations involving `char`s, what is the most reasonable value for `sizeof(char)`?

Problem 10: (12 points)

For 1.5 points each, answer the following questions, all related to UNIX.

(1) How can we quickly identify if a path is absolute or relative?

(2) Describe in English the file names that would be matched by the bash wildcard pattern `??[abc]??`

(3) Same as previous question but for the pattern `T???*.?`

(4) What is one of the three files that bash always looks for when it is started as a login shell?

(5) The command "`ln -s $fall15/a6 a6`" creates an entry named `a6`. In two words or less, what is `a6`?

(6) What does bash's `PATH` variable specify?

(7) Fill in the blank to make `wc` produce the output shown, regardless of the current directory contents.

```
% _______________ | wc
      0       0       0
```

(8) Imagine that one of `whm`'s C slides indicates that an example is in `switch2.c`. Exactly what command(s) would you type on lectura to see the contents of `switch2.c`? Assume your current configuration on lectura with `.bashrc`, symlinks, etc.

Extra Credit Section (½ point each unless otherwise noted)

(1) The C11 standard mentions "translation unit" in a number of places. What is a translation unit?

(2) Using C's `while`, write an infinite loop in as few characters as possible.

(3) Write a version of `mvqt` (page 4) that handles names like `smith-jones-questions.txt`.

(4) Who is widely recognized as "The Father of ASCII"?

(5) What is the "A1B2 Challenge"?

(6) Tell me something about Lester Moore.

(7) Instead of doing "`gcc x.c && a.out`", a friend is doing "`gcc x.c | a.out`". What's a subtle problem your friend may encounter?

(8) Who was the featured speaker at the College of Science Honors Convocation, on Friday, Oct 9, 2015 <u>or</u>, at what time did the convocation start?