CSE 303 Midterm Exam

May 5, 2008

Name ________Sample Solution_________

The exam is closed book, except that you may have a single page of hand-written notes for reference.

If you don't remember the details of how a specific function or command works (parameter lists, whether an error is indicated by a non-zero return code or null pointer, etc.), write down the assumptions you need to make and as long as they're reasonable you're ok.

If you have questions during the exam, raise your hand and someone will come to you. Don't leave your seat.

Please wait to turn the page until everyone has their exam and you have been told to begin.

Advice: The solutions to many of the problems are quite short. Don't be alarmed if there is a lot more room on the page than you actually need for your answer.

More gratuitous advice: Be sure to get to all the questions. If you find you are spending a lot of time on a question, move on and try other ones, then come back to the question that was taking the time.

1	/ 14
2	/ 12
3	/ 15
4	/ 14
5	/ 14
6	/ 15
7	/ 16
Total	/ 100

Question 1. (14 points) Suppose the following files and subdirectories exist in a directory:

```
.bashrc                       proj/test.exe
.emacs                        proj/test.c
.bash_profile                 proj/test.o
proj                          proj/thing.c
proj/data                     proj/thing.h
proj/data/dict.txt            proj/thing.o
proj/data/smalldict.txt
proj/notes
proj/notes/todo.txt
proj/notes/readme.txt
```

Answer the following questions assuming that this directory is the initial current directory when each of the following sets of commands are executed.

(a) What output is produced by the following commands?

```
cd  proj
ls  *.[ch] > xyzzy
ls  notes/* >> xyzzy
sort  xyzzy
```

```
notes/readme.txt
notes/todo.txt
test.c
thing.c
thing.h
```

Many people missed that "`notes/`" is part of the output of the second `ls` command; we only deducted a point for that.

(b) What do the following commands do?

```
cd  proj
mv  */*.txt  ..
```

Moves the files in the `notes` and `data` subdirectories whose names end in "`.txt`" to the parent directory above `proj`.

Question 2. (12 points) Your instructor has never been able to break the habit of using the `more` command to page output written to the terminal, even though he knows he should use `less` instead. Describe **two different** ways of setting up his Linux environment so that whenever he uses the command

```
more <list of arguments>
```

the command

```
less <list of arguments>
```

is executed instead. For full credit, your answer should give specific details (code, bash commands, or whatever is needed), not just general ideas.

The two basic answers were:

- **Add an alias command to `.bashrc`, `.bash_profile`, or other appropriate file containing**

  ```
  alias more=less
  ```

- **Create a shell script named `more` in some directory on the user's search path before the directory containing the regular `more` command with the following code**

  ```
  #!/bin/bash
  less $@
  ```

Another good solution would be to place a link named `more`, linked to `/usr/bin/less`, in some directory that appears on the user's search path before `/usr/bin`.

A not-good solution suggested by some people would be to remove `more` from `/usr/bin`, or replace it with `less`, or otherwise alter the system directories. Since a normal user doesn't have permission to do this on an arbitrary system like `attu`, this isn't a possible solution in general, while the ones listed above are.

Question 3. (15 points) Give the code for a bash shell script that makes copies of all of the simple files that are given as its arguments. The file copies should have the same names as the original files but with the additional file extension ".bak" at the end. The script should ignore any arguments that are directories (i.e., only copy files that are ordinary files). If an error occurs (say a file doesn't exist, or the copy operation doesn't succeed), ignore it and continue processing the remaining arguments.

Example: Suppose the current directory contains the following files and directories:

```
file1  doc/file2  file3.txt  file4.bak
```

If the script is run in this directory, it should copy `file1`, `file3.txt` and `file4.bak` to create the following additional files:

```
file1.bak  file3.txt.bak   file4.bak.bak
```

The subdirectory `doc` should be skipped.

Hint: In the test command, `-d` can be used to test for directories, and `-f` to test for files.

```
#!/bin/bash
while [ $# -ge 1 ]
do
     if [ -f ${1} ]
          cp ${1} ${1}.bak
     fi
     shift
done
```

Question 4. (14 points) Give a **sed** command that processes a file containing URLs and other text, and extracts information about the web domains that appear in URLs in the file. In particular, we want to find all of the URLs beginning with http:// and extract the domain kind (org, com, or edu) and the domain name (washington, microsoft, yahoo, etc.), and write these to a separate file.

Suppose that the input file contains the following URLs somewhere in the input lines:

. . . http://www.cs.washington.edu . . .
. . . http://download.research.microsoft.com . . .
. . . http://goodies.123freestuff.org . . .
. . . http://washington.edu . . .

The output produced by sed for these lines should contain the following, with a blank space between the domain kind and the domain name.

```
edu  washington
com  microsoft
org  123freestuff
edu  washington
```

You can make the following assumptions in your solution:

- Each line in the input file contains at most one http:// URL. We are only interested in URLs that begin with http://, and we are not interested in input lines that do not contain a http:// URL.
- You can assume that all URLs end with one of .com, .org, or .edu
- You can assume that the domain name (the word immediately preceding .org, .com, or .edu) consists only of lower-case letters and digits.

Write your **sed** command to process the file `urldata` and produce a file named `domains`, where the information in the `domains` file is extracted from `urldata` as described above.

```
sed  's/http:\/\/.*\([a-z0-9]+\)\.\(com|edu|org\)/\2  \1/'  urldata
       > domains
```

The key points above are to be sure to capture the domain name and kind in `\(  \)` parentheses, and it's important to have a literal period `\.` between them.

The pattern can be a little simpler if some delimiter other than / is used, say _ :

```
's_http://.*\([a-z0-9]+\)\.\(com|edu|org\)_\2 \1_'
```

But you can't use + or | for the delimiters without dealing with the fact that these are used as operators in the regular expression.

Question 5. (14 points) The following function is, as explained in the comments, supposed to return a newly allocated string that contains a copy of the string that is its argument.

(Corrections shown below in green, and with strikeouts of the original problems.)

```
#include <string.h>
#include <stdlib.h>

/* return a newly allocated string containing a copy of the contents of */
/* string s.  Pre: assume s is a properly-formed C string.              */

~~void~~ char * clone(char * s) {    /* incorrect result type */

  char* copy;

  int len, k;

  len = strlen(s);

  copy = (char *)malloc((len+1)*sizeof(char));  /* missing cast and +1 */

  for (k = 0; k < len; k++) {

    copy[k] = s[k];

  }

  copy[len] = '\0';   /* need \0 at end of copied string */

  ~~free(s);~~            /* error to free original string */

  return copy;

}
```

There was at least one other common way to fix the "no `\0` at the end of the string" bug: change the loop upper bound from `k<len` to `k<=len`. But it was not ok to, for example, replace the loop with a `strcpy` operation, since the instructions said to fix the existing code and not replace it. Changing the code to use pointers everywhere instead of subscripted arrays was also trouble for the same reason.

Unfortunately, there are bugs in this function. Indicate the problems in the above code and describe how to fix them by writing in corrections, adding code if needed, or crossing out incorrect code and showing how to fix it.

For full credit you should fix the existing code and not rewrite it to do the required work in a different way.

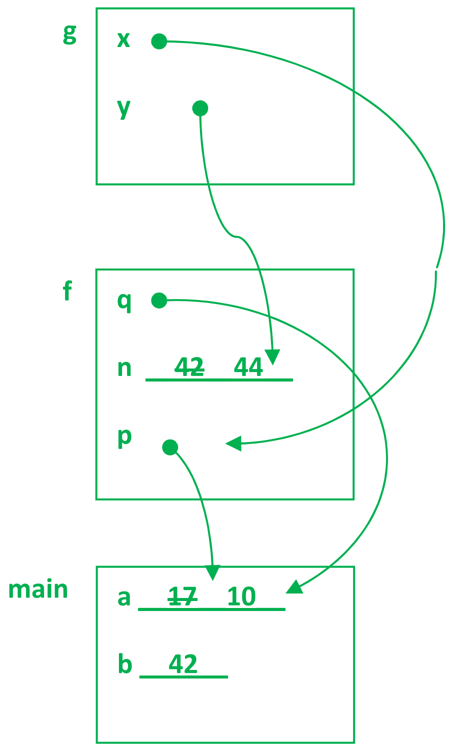
Question 6. (15 points) Consider the following C program:

```
#include <stdio.h>

void g(int **x, int *y) {
  **x = 10;
  printf("g: %d %d\n", **x, *y);
}

void f(int *q, int n) {
  int *p;
  n = n+2;
  p = q;
  g(&p, &n);
  printf("f: %d %d %d\n", *p, *q, n);
}

int main() {
  int a = 17;
  int b = 42;
  f(&a, b);
  printf("main: %d %d\n", a, b);
  return 0;
}
```



(a) Draw a diagram with boxes and arrows showing the situation right at the point when execution reaches the `printf` statement in function `g`. Be sure to show the values of all variables in all active functions. If a variable is a pointer to another variable, show its value by drawing an arrow between the variable name and the storage location (variable) it points to. Draw your diagram above, either to the right of, or below, the code.

(b) What output is produced when this program is executed?

```
g:  10  44
f:  10  10  44
main:  10  42
```

Question 7. (16 points) Complete the definition of the following function so that it alters its string argument by converting the first character of each word to upper case. For example, if the original string is

how now, brown   cow? Eh?

the string should contain the following after the function executes:

How Now, Brown   Cow? Eh?

More specifically, the function should convert the first non-space character in the string to upper case, and should convert to upper case each additional non-space character that follows one or more space characters (blanks, tabs, etc.). All other characters should remain unchanged. The function should alter the characters in the original string and should not create a new string.

Potentially useful information: the functions `toupper(ch)` and `tolower(ch)` in `ctype.h` return the upper- and lower-case versions of their character arguments if the argument has the opposite case, otherwise they return the original character. The function `isspace(ch)` in `ctype.h` returns true (1) if `ch` is a space character (blank, tab, etc.), otherwise it returns false (0).

```
#include <ctype.h>
#include <string.h>

/* Change the words in s so they start with uppercase characters */
void capitalize(char * s) {

  char *p;          /* next unprocessed character in s */
  int convert_next; /*"next non-space character should be converted"*/
  p = s;
  convert_next = 1;
  while (*p != '\0') {
    if (isspace(*p)) {
      covert_next=1;
    } else {
      if (convert_next) {
        *p = toupper(*p);
        convert_next = 0;
      }
    }
    p++;
  }
}
```