

CSE 374 Midterm Exam 5/04/09

Name _____

Do **not** write your id number or any other confidential information on this page.

There are 7 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

You may have a sheet of hand-written notes if you brought them. Other than that, the exam is closed book, closed notes, closed laptops, closed twitter, closed telepathy, etc.

Please wait to turn the page until everyone is told to begin.

Score _____ / 100

1. _____ / 15

2. _____ / 15

3. _____ / 5

4. _____ / 10

5. _____ / 20

6. _____ / 15

7. _____ / 20

Question 1. (15 points) Suppose we have the following directories and files stored in a Linux system. Directories are indicated by names ending with a slash ('/'), and the contents of each directory appear below it indented a few spaces.

```

cse374/
  index.html
  syllabus.pdf
  lectures/
    01-intro.ppt
    01.intro.pdf
    02-shell.ppt
    02-shell.pdf
    03-scripts.ppt
    03-scripts.pdf
    lec1.history
    lec2.history
    lec3files.tar

```

```

cse374/ (continued)
  homework/
    hw1.html
    hw2.html
    hw3.html
    hw3files/
      sample1.txt
      sample2.txt
  exams/
    midterm.doc
    midterm.pdf

```

What happens when we enter each of the following sequences of commands? Assume that each part of the problem is completely independent, that the files and directories are as given above when each group of commands is executed, and that the current working directory is initially the `cse374` top-level directory. You should describe the output displayed or the effect produced. You do not need to describe the detailed effects of each of the individual commands. For instance, if the commands are “`cd exams; rm *`”, a good answer would be “delete the files in the exams directory”.

- (a) `cd homework; echo *.htm*`
- (b) `cd homework; mv */*.txt ..`
- (c) `cd lectures; cd ../ex*; ls`

Question 2. (15 points) This question involves the same set of files and directories as question 1. The list is repeated below for convenience.

```
cse374/
index.html
syllabus.pdf
lectures/
  01-intro.ppt
  01.intro.pdf
  02-shell.ppt
  02-shell.pdf
  03-scripts.ppt
  03-scripts.pdf
  lec1.history
  lec2.history
  lec3files.tar

cse374/ (continued)
homework/
  hw1.html
  hw2.html
  hw3.html
  hw3files/
    sample1.txt
    sample2.txt
  exams/
    midterm.doc
    midterm.pdf
```

Write `bash` commands to perform the following tasks. You should assume that each part of the question is executed independently of the other parts, each part starts with the original set of files and directories, and the initial working directory is `cse374`.

(a) Create a new subdirectory of `cse374` named `pdfs`. Then use a single `mv` command to move all the `pdf` files (files whose names end in `.pdf`) shown above into that new subdirectory. Your `mv` command must use wildcards (`*.pdf`) to specify the files – you may not write out the individual `pdf` file names.

(b) Remove the `homework` directory and all of its contents with a single command.

(c) Write to `stdout` the number of files in the `lectures` directory that have names ending in `.ppt`. The output should just be a number with no titles, labels, or other text. (The answer is 3 in this case, but your answer must actually count the files – “`echo 3`” is not appropriate. You will receive substantial partial credit if your command(s) print the correct answer but with some additional text.)

Question 3. (5 points) Several of the students in the class are doing their homework on dante, and it is tedious to enter the full `ssh dante.u.washington.edu` command to log on repeatedly.

Give a bash `alias` command that a user could include in their `.bash_profile` file so that typing

```
dante
```

would execute the above `ssh` command to start a remote login session on dante.

Question 4. (10 points) Write regular expressions that could be used with `grep` to locate words with the following properties in a file containing a list of words, one word per line (like the one demonstrated in class).

(a) All words that start with the letter 'z' and end with any letter that is a vowel (i.e, end with one of the letters 'a', 'e', 'i', 'o', or 'u').

(b) All words that contain a three-letter sequence that appears at least three times. (For example, *expressionlessness* – *ess* appears three times – or *hemidemisemiquaver*, which has 3 occurrences of *emi*.)

Question 5. (20 points) `gcc` messages have a specific format. Each message contains the name of the file, the line number where a problem was detected, the word “error” or “warning”, and a description of the problem. There are colons separating the parts of the messages, and sometimes, but not always, there are spaces either before or after the colons. Here is a brief example:

```
foo.c:7: error: 'n' undeclared (first use in this function)
foo.c:14: error: parse error before 'g'
error.c:12: warning: control reaches end of non-void function
bar.c:5: error: 'ch' undeclared (first use in this function)
bar.c:35: warning: implicit declaration of function 'malloc'
```

For this problem, write a bash script that reads a file containing `gcc` messages in this format and writes to standard output a sorted list of every file name that appears in one or more error messages. Files named in warning messages should only be included if they also appear in an error message. Each file name should appear only once in the output, and the list of file names should be sorted alphabetically. For example, if the script is run on a file containing the above messages, the output should be:

```
bar.c
foo.c
```

(`error.c` does not appear in the list because that file name only appears in a warning message.)

If the script is given no arguments or if there is more than one argument, or if the argument does not specify an existing file (not a directory), the script should terminate with an appropriate error message written to `stdout`. If the file does exist, you may assume that it is a text file containing properly formatted `gcc` messages and you do not need to check for further errors.

Hints: `grep`, `sed`, `sort`, and `uniq` (`uniq` copies its `stdin` to `stdout`, but if two or more immediately consecutive lines in the input are the same, only the first of the identical lines in that group is copied to `stdout`.)

Write your answer in the space provided on the next page.

Question 5. (continued) Write your answer on this page. The sample messages from the previous page are repeated below for reference (and perhaps inspiration)

```
foo.c:7: error: 'n' undeclared (first use in this function)
foo.c:14: error: parse error before 'g'
error.c:12: warning: control reaches end of non-void function
bar.c:5: error: 'ch' undeclared (first use in this function)
bar.c:35: warning: implicit declaration of function 'malloc'
```

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

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

int main() {
    char *s = (char*)malloc(12*sizeof(char));
    char *t = (char*)malloc(15*sizeof(char));
    strcpy(s, "good");
    strcpy(t, "hi there");
    char *p = t+4;
    t = s;
    strcat(s, "bye");
    printf("s = %s\n", s);
    printf("t = %s\n", t);
    printf("p = %s\n", p);
    return 0;
}
```

(a) What output does this program produce when it is executed? (It does execute successfully.) Feel free to draw diagrams showing memory to help answer the question and to help us award partial credit if needed.

(b) One of your colleagues noticed that this program allocates memory without freeing it, and proposes that the following code be added right before `return 0;` to fix this.

```
free(s);
free(t);
```

Once this code is added, does it work and free the dynamic storage properly? Why or why not?

Question 7. (20 points) (The small C programming exercise.) For this problem, complete the definition of function `main` below so that it prints `yes` if the string `xyzzzy` appears as an argument on the command line that executes the program and prints `no` if not. The argument list must contain the string `xyzzzy` exactly as a single argument for the program to print `yes`; if it is not an exact match the program should print `no`. For example, if the program is run with this command line

```
./a.out there is a xyzzzy here
```

the program should print `yes`. If the program is run with

```
./a.out this "contains xyzzzy" but doesn't match exactly
```

the program should print `no`. Add your code to the main program below. (Notice that this program should *only* examine its argument list. It should *not* read any data files.) Assume that any necessary library header files are already included. You don't need to write `#includes`.

```
int main(int argc, char ** argv) {
```

```
}
```