CSE 374 Midterm Exam 5/3/2019

Name ____ Id # _______________

There are 6 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.

The exam is closed book, closed notes, closed electronics, etc.

Many of the questions have short solutions, even if the question is somewhat long. Don't be alarmed.

If you don't remember the exact syntax of some command or the format of a command's output, make the best attempt you can. We will make allowances when grading.

Relax, you are here to learn.

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

Score _________________ / 100

1.	______ / 10	Linux Commands
2.	______ / 6	Aliases
3.	______ / 6	Getting Help
4.	______ / 22	Scripting
5.	______ / 16	Grep/sed
6.	______ / 16	C analysis
7.	______ / 24	C programming
8.	______/ 2	Extra Credit

The **last page** of the exam contains reference information that may be useful while answering some of the questions. Do not write on this page – it will not be examined.

CSE 374 Midterm Exam   5/3/2019

Question 1. (10 points as 2,2,2,4 pts.) **Linux commands**. Here is a brief transcript from a Linux terminal session (user input follows each `$` prompt, the rest is system output):

```
$ pwd
/home/bashexpert

$ ls -R
.:
374files  csehome  cutilities  Desktop  bin  TRASH

./374files:
hw3  hw4  lec10  lec13  lec3  lec5  lec7  lec9  hw3.tar
hw5  lec11  lec2  lec4  lec6  lec8  welcome.message

./374files/hw3:
cdf.eps  popular.html  runexperiment.sh  myurls.txt
popular-small.txt  runexperiment.sh  produce-cdf.gnuplot
scatterplot.eps  parse.sh produce-scatterplot.gnuplot
testurls  perform-measurement.sh  results.txt

./374files/hw4:
gasp  gasp.c  gasptest.c  newout

./374files/lec10:
Argumentdemo  argumentdemo.c  dangling  dangling.c
```

< … truncated for space …>

```
./scripts:
clean  fibo  msdel

./TRASH:
gasp.c~.tar Makefile~.tar
```

Based on the above, please answer the following prompts:

Write a command to see what is in the '`welcome.message`' file while in your current directory:

CSE 374 Midterm Exam   5/3/2019

Write a command to determine which of the scripts in your bin folder are executable:

Write a command to remove all the ‘`tar`’ files from TRASH:

Most of the items in 374files are directories. You run the command:

```
$ wc `ls -R 374files/*/*.c`
```

Please describe what this command will do. Be specific about which files will be considered, and what the output of the command is.
Hint, describe what the input to `wc` is, and then what `wc` creates.

Question 2. (6 points, 3 each) **Aliases**. Give `alias` commands that will create aliases that work as described below.

(a) Define an alias `mkex` that allows you to put personal scripts to use. This alias will make the input file(s) executable for the user and all group members. You will use the alias with `$mkex newscriptfile`.

(b) You have run `python clint.py` on your new C file and received the error message "Line ends in whitespace". This happens frequently, so you decided to make an alias (`rmws`) that will automatically run your sed command (`sed 's/\s*$//g'`) to remove it. Assuming you creates the alias `rmws` correctly to run the precise sed command above, please show how you would use this command to process a file newcode.c and save the updated code in a file called `newcode2.c`.

Question 3. (8 points) **Getting help**.

You know there is a command that can give you information about the 'network interface'. What can you type at your prompt to find that command?

You discovered that `ifconfig` will probably do the trick. List THREE things that you could do to understand `ifconfig` and how to use it. If something involves the command line, be specific about exactly what commands you would use.

Question 4. (22 points) **Scripting:**

What is the difference between using `source` with a file, and calling the file as an executable?

Write a shell script called ‘`worknew`’ that takes a file name as the first command-line argument, followed by zero or more directories. The script will print the number of recently (in the past week) modified files in a directory to the file given as the first argument.

- If no arguments are provided, print an appropriate error message to `stderr` (stream 2) and exit with return code 1. Otherwise exit with return code 0.
- You should check the directories for existence; if a directory is not a valid directory print an error message to `stderr` and continue to the next argument.
- Your script should handle file names that have embedded blanks in them like “output file”.
- If no valid directories are provided the output file will be empty.

For example, if you execute the following command with your script `worknew.sh`, the file `counts.txt` should contain one line with the number of files in lec11 that have been modified in the last week. Having multiple directories on the input should produce a multi-line output file.

```
$ ./worknew.sh counts.txt ~/374files/lec11
$ more counts.txt
/home/youruserid/374files/lec11: 3
```

Hints:

- There may be some useful info on the last page of this exam.
- You know that the command [`find` *`dirname`* `-type f -mtime -7`] will produce an output of all the files that have been modified in the past week.
- Counting the number of lines in the output should provide the correct value for your report.
- You will get partial credit: if you think you are doing something confusing, consider commenting it with ‘#’.
- You will not be penalized for white-space or (lack-of) comments on this question.

Write your answer below. The `#!/bin/bash` that starts the script is provided for you.

```
#!/bin/bash
```

Question 5. (16 points: 4,4,8pts each) (`grep/sed`)

You run the following command in your 'home' directory: `grep 'alias' .bash*`
What do you expect to find?

You were pleased that your alias from earlier in the test worked. Explain each section of your sed command: `sed 's/\s*$//g'`

You have decided to change your name. You know that all of your .c files have a line denoting authorship which looks like:

```
/*  filename, authored by Your Name possibly something else
```

Write a sed command that will replace each instance of `Your Name` in the above context with `New Name`, in every .c file in the current directory.
Hint – you may collect all the non-name text in backreferences to be used in the replacement string.

Question 6. (16 points) Analyze a C program. As is usual, this program compiles and executes without warnings or errors. Then, answer the questions on the following page.

```
#include <stdio.h>

void functionB( int *y, char *z ) {
  *y = 100;
  z[0]=z[1];
  printf("y=%d, z=%s\n", *y, z);
}

void functionA( int *w, char *x )    {
  int temp = *w;
  functionB( w, x+1 );          /* Note this!! */
  *w = *w+temp;
  printf("w=%d, x=%s\n", *w, x);
}

int main ( void ) {
  int a = 77;
  char b[] = "cat";

  printf("a=%d, b=%s\n", a, b);
  functionA( &a, b );
  printf("a=%d, b=%s\n", a, b);

  return (1);
}
```

This code prints four lines. Please write these lines, precisely as they are printed, in order below. If you would like, use the space on page 10 to draw a diagram showing the contents of memory as the program executes.

Printf1: ______________________________________

Printf2: ______________________________________

Printf3: ______________________________________

Printf4: ______________________________________

Question 6. Continued (4pts):

You wish to add a return statement to functionA:

```
  return &temp;
```

Assuming you change the return type, and then use the value in main, what other issues are there with taking this step? How could you fix them?

You have run the program, but still don't know why the answer is what is. Please describe something you can do using **gdb** to help understand what value is in each variable at any given time.

Question 6. (optional) Draw your diagram here showing the contents of memory during execution of the mystery program.

Question 7. (24 points) C programming.

You are working on a program to read in data about people, and perform some analysis on your set of data. In this problem, you will be slowly building functionality towards the end goal of having a linked list of people who can be sorted on various parameters.

The set up for the code is here:

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

#define MAX_NAME 25

// define a struct person here

person* makePerson (char *name, int a);
int orderPeople (person *p1, person *p2);

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

}
```

(a) (4 points) First, you will need to define your struct that stores a single person. For now, your records will include a name, and an age. Please notice that a MAX_NAME was created to set a length limit for the name.

(b) (8 points) Now that you have your struct, write the two supporting functions that have forward declarations above – one to make a new person, and one to sort two people. For now, 'orderPeople' will return '1' if p1's name comes first alphabetically, and '2' otherwise.

```
person* makePerson (char *name, int a) {
```

```
int orderPeople (person *p1, person *p2){
```

Question 7. (cont.) (8 pts) Now, write the main program that uses the file given as an argument to the program (i.e., `argv[1]`). In this file you will find lines of text with a name, followed by a space, followed by an age:

```
Karel 30
Bee 42
```

Of course, you will want to do some error checking. You will also want to exit with an error message if anything goes wrong. You will want to use good file-handling and string manipulation practices. If everything goes right you will want to do the following:

1. For two people
   a. Get the name, age pair from the input file
   b. Create a new person with that data
2. Figure out what order the two people are in
3. Print a line of output with the result of (2)

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

Question 7. (cont.) (4 pts)

You are concerned that you may have created a memory leak in your code. Before you move on to add more functionality, what steps can you take to examine your heap usage? Please put the command you might use at a prompt:

The first time you run your program it produces a seg-fault. What gdb command can you use to determine the state of the code when this occurs?

Extra Credit (2 pts)

What do the people in the following list have in common?
Mark D, Clarence E, Philip E, Timnit G, Grace H, Margaret H, Katherine J, Ada L, Jerry L, Ellen O, Alan T, Jeannette W.

CSE 374 Midterm Exam   5/3/2019

Reference Information

Some of this information might be useful while answering questions on the exam. **Do not write on this page – anything written here will not be graded.**

Shell: Some of the tests that can appear in a [ ] or [[ ]] test command in a bash script:

- string comparisons: =, `!=`
- numeric comparisons: `-eq`, `-ne`, `-gt`, `-ge`, `-lt`, `-le`
- `-d` *name* test for directory
- `-f` *name* test for regular file

Shell variables: `$#` (# arguments), `$?` (last command result), `$@`, `$*` (all arguments), `$0`, `$1`, … (specific arguments), `shift` (discard first argument)
Redirect with `>`, `>>`. '`&1`' is the location of stdout, '`&2`' is the location of stderr.
Create an empty file: use 'touch'
Execute a command in the stream: use backquotes, or pipe (there is a difference)

Strings and characters (`<string.h>`, `<ctype.h>`) Some of the string library functions:

- `char* strncpy`(*dest*, *src*, *n*), copies exactly *n* characters from *src* to *dst*, adding '\0's at end if fewer than *n* characters in *src* so that *n* chars. are copied.
- `char* strcpy`(*dest*, *src*)
- `char* strncat`(*dest*, *src, n*), append up to *n* characters from *src* to the end of *dest*, put '\0' at end, either copy from *src* or added if no '\0' in copied part of *src*.
- `char* strcat`(*dest*, *src*)
- `int   strncmp`(*string1*, *string2*, *n*), <0, =0, >0 if compare <, =, >
- `int   strcmp`(*string1*, *string2*)
- `char* strstr`(*string, search_string*)
- `int   strnlen`(*s, max_length*)
- `int   strlen`(*s*)
- Character tests: `isupper`(*c*), `islower`(*c*), `isdigit`(*c*), `isspace`(*c*)
- Character conversions: `toupper`(*c*), `tolower`(*c*)

Files (`<stdio.h>`)

- Default streams: `stdin`, `stdout`, and `stderr`.
- `FILE* fopen`(*filename*, *mode*), modes include "r" and "w"
- `char* fgets`(*line, max_length, file*), returns `NULL` on end of file
- `int fscanf`(*FILE* stream, const char* format, …*), reads text with printf formatting. Additional arguments are pointers to the requested data types.
- `int   feof`(*file*), returns non-zero if end of *file* has been reached
- `int   fputs`(*line, file*)
- `int   fclose`(*file*)
- A few `printf` format codes: `%d` (integer), `%c` (char), `%s` (char*)