CSE 374 Midterm Exam 4/27/18

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.	_____ / 8	Linux Commands
2.	_____ / 8	Aliases
3.	_____ / 6	Getting Help
4.	_____ / 20	Scripting
5.	_____ / 16	sed
6.	_____ / 18	C analysis
7.	_____ / 24	C programming

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 while grading. You may remove that page from the exam if you wish.

CSE 374 Midterm Exam 4/27/18

Question 1. (8 points, 2 each) Linux commands. Here is a brief transcript from a Linux terminal session (user input follows each `$` prompt, the rest is system output):

```
$ cd
$ pwd
/homes/astudent
$ ls -l
-rw-rw---- 1 astudent   10 Apr 27 09:32 a.txt
-rw-rw---- 1 astudent    8 Apr 27 09:32 b.txt
drwxrwxr-x 2 astudent 4096 Apr 27 09:38 temp
$ cd temp
$ ls
anotherdir  a.txt.tmp  b.txt.tmp  nottmp.txt
$ cd ..
$ mv ~/b.txt temp
```

After each of the following commands, write the output that it produces or give a command to achieve the goal. You should assume that the first command (part a) is executed immediately after the above commands and each subsequent line of commands is executed after the commands in the previous parts of the question have been executed (i.e., the commands in parts (a) through (d) are run one after the other).

(a) `x=* ; echo $x > e.txt ; cat e.txt`

`a.txt temp`

(b) `whoami`

`astudent`

(c) Write a command to make the file `b.txt` readable by everyone.

`chmod +r temp/b.txt`

(d) Write a command to remove all temporary files (which end with the extension `.tmp`) from the `temp` directory and all its child directories. Only remove `.tmp` files.

`rm temp/*.tmp temp/*/*.tmp`
OR
`rm temp/*.tmp temp/anotherdir/*.tmp`

(rm –r does not do the right thing here – that will remove child directories, but not look for child directories that match the pattern)

Question 2. (8 points, 4 each) Aliases. Give `alias` commands that will create aliases that work as described below.

(a) Define an alias `edit` that will allow you to easily edit your `.bashrc` file. The alias should open `emacs` to allow you to edit `.bashrc`, and then call `source` on it to load/execute it. Remember that `.bashrc` is located in your home directory.

```
alias edit='emacs ~/.bashrc ; source ~/.bashrc'
```

(b) Define an alias `...` (three dots) that moves you into the grandparent directory of your current working directory (i.e. the grandparent is the parent directory of the parent directory of the working directory).

```
alias ...='cd ../..'
```

Question 3. (6 points) Getting help. You are told that you will need to use the program `cal` to implement your homework assignment. List THREE things that you could do to understand `cal` and how to use it. If something involves the command line, be specific about exactly what commands you would use.

```
1) man cal
2) cal --help
3) Online Linux man pages
4) course resource links
5) Linux pocket guide
6) search Google or StackOverflow for "cal bash usage"
(there are potentially other answers here, although
 "Google" or "internet search" isnt an adequate answer)
```

Question 4. (20 points) (Scripting) Write a shell script that takes a file name as the first command-line argument, followed by zero or more integers. The script should calculate the sum of the provided integers and print the sum to the file given as the first argument.

- If an integer is less than 0, you should add its absolute value to the sum (i.e. if the integer is -4, you should add 4 to the sum).
- 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.
- Your script should handle file names that have embedded blanks in them like "output file".
- If no integers are provided, the sum is 0.

For example, if you execute the following command with your script `sum.sh`, the file `sum.txt` should store the single value "8" (which is 1+3+0+4).

```
$ ./sum.sh sum.txt 1 3 0 -4
```

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

```
#!/bin/bash

if [[ $# -lt 1 ]]
then
  echo "$0: please enter at least one argument" >&2
  exit 1
fi

output="$1"
sum=0
while [[ $# -gt 1 ]]
do
  shift
  val=$1
  if [[ $val -lt 0 ]]
  then
    ((sum=sum-val))
  else
    ((sum=sum+val))
  fi
done

echo $sum > "$output"
```

Question 5. (16 points) (`grep/sed`) You may have noticed that on the course website, some email addresses are written with "[at]" instead of "@", which is to prevent scripts from easily identifying personal email addresses and sending spam email.

```
Contacts for CSE 374:
mwinst[at]gmail.com (Instructor)
cse374-staff@cs.washington.edu (for staff - prefer this one!)
For grading questions, email your TA
ta[at]cs.uw.edu
```

Since you now know regular expressions, you want to write a command to extract all email addresses from the file, even if they have "[at]" instead of "@".

Give a single Linux command line that will print all email addresses from the file `input` (and ONLY the email addresses), with any "[at]" replaced with "@".

If the contents of the file `input` are as above, the output of your command should be:

```
mwinst@gmail.com
cse374-staff@cs.washington.edu
ta@cs.uw.edu
```

You should assume that for this input file, any email addresses will be the first thing on a line. An email address can contain any character except a space. You should also include emails that already have "@" in them.

Hint: your command will probably have more than one command connected by a pipe (|).

Note: if the handwritten solution won't fit on one line, just continue on a second line at some obvious place, like right before or after any pipe (|) symbol. You may use "-E" for extended regular expressions if you prefer.

```
grep '^[^ ]\+\(@\|\[at\]\)' emails.txt |
  sed 's/^\([^ ]\+\)\(@\|\[at\]\)\([^ ]\+\).*/\1@\3/'

or with extended regular expressions:

grep -E '^[^ ]+(@|\[at\])' emails.txt |
  sed -E 's/^([^ ]+)(@|\[at\])([^ ]+).*/\1@\3/'
```

Question 6. (18 points) The traditional, annoying C program. As is usual, this program compiles and executes without warnings or errors.

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

void bar(int *a, int b, int *c, char *d) {
  c[1] = *a + b;
  d[*c] = d[2];
  b = 17;
  int *r = c + 1;
  c = a;
  *c = *a + 1;
  printf("during: %d %d %d %d %s\n", *a, b, *c, *r, &d[1]);
}

int main(int argc, char **argv) {
  char cat[7] = "dog";
  int x = strlen(cat);
  int z[3];
  z[0] = 1;
  z[1] = 2;
  z[2] = 4;
  int *p = &x;
  int *q = &z[1];
  q[1] = *p + 10;
  printf("before: %d %d %d %d %s\n", z[0], z[1], z[2], x, cat);
  bar(p, x, q, cat);
  printf("after: %d %d %d %d %s\n", z[0], z[1], z[2], x, cat);
  return 0;
}
```

Fill in the lines below to show the output produced when this program is executed. If you would like, use the next page to draw a diagram showing the contents of memory as the program executes. This will help us to give you partial credit if the answer you give is incorrect. Show boxes for the local variables and parameters of all active functions.

before: **1 2 13 3 dog**

during: **4 17 4 6 og**

after: **1 2 6 4 dog**

Question 7. (24 points) C programming.

"Pig Latin" is a language transformation in which the first character of a word is moved to the end and "ay" is added:

"banana"	=> "ananabay"	"dog"	=> "ogday"
"cat"	=> "atcay"	"pool"	=> "oolpay"

If a word already starts with a vowel, then "ay" is added but the first character is not moved to the end:

"aardvark"	=> "aardvarkay"	"ear"	=> "earay"

Write a C program that will read a text file that contains one word per line and print to `stdout` the Pig Latin forms of the words, one per line. You can assume all words are entirely lowercase.

You should use standard C library functions in your solution when appropriate (the last page of the exam contains a summary of some that might be useful). You do not need to write `#include` directives – assume this has been done for you.

(a) (8 points) Complete the function `pig(s)` below so it modifies the string parameter to store the Pig Latin form of the word. The string `s` is a proper C string – an array of characters with a '\0' byte at the end – and `pig` can assume that it has extra allocated space for the two "ay" characters to be appended. Hint: a vowel is 'a', 'e', 'i', 'o', or 'u'.

```
// Modifies the given string into Pig Latin form.
// Assumes s has extra allocated space for 2 more chars.
void pig(char *s) {
  char ch = s[0];
  if (ch != 'a' && ch != 'e' && ch != 'i' &&
      ch != 'o' && ch != 'u') {
    int length = strlen(s);
    strncpy(s, s + 1, length - 1);
    s[length - 1] = ch;
  }
  // Add "ay" to the string.
  strncat(s, "ay", 2);
}
```

(continued on next page)

Question 7. (cont.) (b) (16 points) Now, write the main program that opens the file given as an argument to the program (i.e., `argv[1]`), reads each line in that file, and prints the Pig Latin form of the word on that line to `stdout`. If the file name is missing, or if the file cannot be opened, the program should print a suitable message to `stderr` and terminate with `EXIT_FAILURE`. If no error is detected, the program should terminate with `EXIT_SUCCESS` when it is done. Your program must use the `pig` function from part(a) to transform words into Pig Latin.

You may assume that no line of the input has more than 100 characters, including any newline ('\n') or '\0' bytes at the end of each line. An appropriate name has been `#defined` for you to use. You may also assume that there is a '\n' newline at the end of the last line in the file if it matters. Hints: be sure that any '\n' characters at the end of input lines are not accidentally included when appending "ay" at the end. The output lines may be more than 100 characters long given that you are adding "ay" to the end. You should not use dynamic allocation (`malloc`/`free`) in your code – it is not needed.

```
#define MAX_LINE_LENGTH 100

int main(int argc, char **argv) {
  if (argc < 2) {
    fprintf(stderr, "need a file argument\n");
    exit(EXIT_FAILURE);
  }
  FILE * f = fopen(argv[1], "r");
  if (f == NULL) {
    fprintf(stderr,
            "Unable to open file \"%s\"\n", argv[1]);
    exit(EXIT_FAILURE);
  }
  // Add one to length to make sure we have room for "ay".
  // Since we are removing the \n character, we only need
  // one extra, not two.
  char line[MAX_LINE_LENGTH + 1];
  while(fgets(line, MAX_LINE_LENGTH, f) != NULL) {
    line[strlen(line)-1] = '\0';
    pig(line);
    printf("%s\n", line);
  }
  fclose(f);
  exit(EXIT_SUCCESS);
}
```