Each question on this exam lists the points available for that question immediately after the question number. These points total 100 for the entire exam. Each of the multiple choice questions will have exactly one correct answer. Any multiple choice question for which you choose more than one alternative will be incorrect. CLEARLY mark your answer, either by writing the appropriate letter to the right of the choices, or somehow marking the chosen answer. It is up to YOU to ensure that I can easily tell what your answer is.

1. (5 pts) Which UNIX command will rename a file?
   (a) ex  
   (b) wc  
   (c) ls  
   (d) ps  
   (e) mv

2. (5 pts) Which of the following possible programming features is NOT available in bash shell programs?
   (a) If statements  
   (b) Reading user input  
   (c) Performing some task for each member of a list  
   (d) While statements  
   (e) None of the above. That is, all of the above features are supported by bash.

3. (5 pts) Which of the following UNIX commands can NOT change the current working directory?
   (a) cd  
   (b) pushd  
   (c) popd  
   (d) pwd  
   (e) none of the above – that is, all of the above can be used to change directories.
4. (5 pts) Which of the following UNIX commands allows the user to search for occurrences of specific strings within files and print them out if/where found?

(a) sort  
(b) grep  
(c) cp  
(d) echo  
(e) ls

5. (10 pts) You are to write a single command that you could type at a Linux command prompt that will meet the following requirements:
Using Linux commands, and possibly using Linux pipes, print the number of files in current working directory.

6. (5 pts) Write a single Linux command that will determine how “busy” the computer is.

7. (12 pts) Describe the function of each of the following Linux/Unix commands in 20 words or less.

(a) source

(b) find

(c) pushd
8. (15 pts) Solve the following three “bit manipulation” problems. (NOTE: The book’s description of floating point format is slightly different than what I presented in class. So, to start off, please summarize the representation that you’ll be using in the first two parts of this question. That way, I can grade your answers based upon the format you’re using, within reason.)

(a) Show the floating point representation for the value 13\(\frac{1}{4}\). Be sure to indicate where each of the 3 parts of your 32-bit string (sign, exponent, mantissa) begin and end.

(b) What does the following string of 32-bits represent as a fractional decimal number. (To make it easier for you to read, I’ve included a space between the sign and exponent part of the representation and a space between the exponent and mantissa part of the representation.)

1 00000100 0101110000000000000000000

(c) Explain why the “hidden bit” of floating point format does not need to be represented.

9. (4 pts) Show a declaration and instantiation of a variable to represent a dynamically allocated string big enough to contain “Mary had a little lamb.” (This means that you’ll need to declare a variable and then use malloc to “create” space for the string.)
10. (15 pts) A hapless C programmer, henceforth known as Hap, wrote the code below to convert his Java bubble sort to C. He was very happy when his code compiled the first time he tried it. With great anticipation, he ran his program, by submitting the following command (a.out is the name of the executable file).

```
./a.out < bubble.input
```

Below is a copy of bubble.input. It is the single line:

```
-3 6 19 33 -13 101 86 33 6 -102
```

Sadly, the output generated was the following:

The unsorted list is:

```
0 0 0 0 0 0 0 0 0 0
```

The sorted list is:

```
0 0 0 0 0 0 0 0 0 0
```

Hap was somewhat dismayed, but undaunted, he dived into his code. After what seemed hours of tortuous study of the code (it was actually only about 15 minutes), Hap found his problem (in main()), and after a few choice words about C, fixed “the” problem (which required a VERY small change to ONE line), recompiled and reran his code and obtained the following output:

The unsorted list is:

```
-3 6 19 33 -13 101 86 33 6 -102
```

The sorted list is:

```
-3 6 19 33 -13 101 86 33 6 -102
```

Hap, however, was not to be denied. After considerable consultation with the local C guru, Hap was able to find a second problem in the function(s) bubble and swap. This time the “fix” took a few lines of code, but Hap was able to finally solve the problem and get sorted output.

YOUR MISSION. Fix the code! Make sure you both fix the code AND indicate why the output was as shown. Make sure you find and fix BOTH errors (the one Hap found himself as well as the one for which the guru assisted).

Hints:
The “second” problem MAY be a little easier for you to find. You might want to look for it first.
The ORIGINAL output (all zeros) could be different if the program was executed on another machine (the problem is the same, just the manifestation could be different).
Neither problem is an “algorithm” problem. The code (if properly translated) would work fine in Java, Hap’s “native” language. The problem was simply Hap’s poor understanding of C.
The Program is actually on the next page.
swap(int x, int y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}

bubble(int A[10])
{
    int i, j;
    for( i = 0; i < 10; i++ )
        for( j = 0; j < 9 - i; j++ )
                swap(A[j],A[j+1]);
}

main()
{
    /* read in a list of random numbers and sort them */
    int A[10];
    int i;
    for( i = 0; i < 10; i++ )
        scanf("%d",&A[i]);
    printf("The unsorted list is:\n");
    for( i = 0; i < 10; i++ )
        printf("%d ",A[i]);
    bubble(A);
    printf("\nThe sorted list is:\n");
    for( i = 0; i < 10; i++ )
        printf("%d ",A[i]);
}
11. (5 pts) Complete the function, strlen, shown below.

```c
int strlen(char *s) // return the number of characters in s
{
}
```

12. (5) Describe how `X[i]` can possibly be interpreted the same as `i[X]` in C.

13. (9 pts) Describe an “application” that would be easier to implement in a bash shell program than in C.

Show your bash program for that application.

Now write a C program for that application.