ECS 30A **Homework #8 (Part B)** **Fall 2008**

Due: December 3.
Written: 4:00pm in 2131 Kemper.
Programs: 11:59pm using handin to cs30a p7 directory.
Filenames: fraction.c, complexarithm.c, sort_str_len_dyn.c, binary_dyn.c

Programming. All programs should be able to compile with no warnings when compiled with the –Wall option. You should put your name(s) in a comment on the first line of each file. The prompts, and output format of each program must match the examples exactly.

(1) Programming problem #1 on page 567. Filename: fraction.c

```
[…@pc…]$ fraction.out
Input fraction (num/den): 1/2
1/2 = 1/2
[…@pc…]$ fraction.out
Input fraction (num/den): 4/16
4/16 = 1/4
[…@pc…]$ fraction.out
Input fraction (num/den): 32/6
32/6 = 16/3
```

(2) Programming problem 1 on page 576. Filename: complexarithm.c
Note: start with the program complexarithm.c, which is available on the class website.

```
[…@pc…]$ complexarithm.out
Enter the real and imaginary parts of a complex number separated by a space
first complex number  > 1 0
second complex number > 3 0
(1.00) + (3.00) = (4.00)
(1.00) – (3.00) = (-2.00)
(1.00) * (3.00) = (3.00)
(1.00) / (3.00) = (0.33)
|(1.00)| = (1.00)
|(3.00)| = (3.00)

[…@pc…]$ complexarithm.out
Enter the real and imaginary parts of a complex number separated by a space
first complex number  > 2 3
second complex number > 1 2
(2.00 + 3.00i) + (1.00 + 2.00i) = (3.00 + 5.00i)
(2.00 + 3.00i) - (1.00 + 2.00i) = (1.00 + 1.00i)
(2.00 + 3.00i) * (1.00 + 2.00i) = (-4.00 + 7.00i)
(2.00 + 3.00i) / (1.00 + 2.00i) = (1.60 – 0.20i)
|(2.00 + 3.00i)| = (3.61)
|(2.00 + 3.00i)| = (2.24)

[…@pc…]$
```

(3) Filename: sort_str_len_dyn.c

In Homework #7, you wrote a program called sort_str_len.c, which orders a list of strings according to string length (shortest to longest). You may start with your own program or the one from our class website. The original sort_str_len.c program stores the list of strings in a two-dimensional array of characters, with each row of array containing one string. Your assignment is to modify the sort_str..c so that the array will be one-dimensional, its elements will be the pointers to dynamically allocated strings. We referred to it as the *dynamic string version of the program sort_str_len.c.*

Using dynamically allocated strings in this program, we can use space more efficiently by allocating the exact number of characters needed to store a string, rather than storing the string in a fixed number of characters as the original program does.

(4) Filename: binary_dyn.c

Write a *dynamic array version* of the program binary.c, which was described in Homework #6. You can start with your own program or the one from class website.