Due: Wednesday, November 12 .
Written: 4:00pm in 2131 Kemper.
Programs: 11:59pm using **handin** to **cs30a p5** directory.
Filenames: reperr.c, trap.c, grade.c, binary.c

Written: #1, #2, #3, #4, #5 on page 360
#1, #2, #3, #4, #5 on page 421

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. To use functions from math.h, you must have –lm on your compile line (that is an "l" as in library) to link with the math library, e.g., gcc –Wall –lm reperr.c

#2. page 361. Filename: reperr.c

```
[…@pc…]$ reperr.out
Adding 2 1/2's gives a result of 1.
Adding 3 1/3's gives a result of 1.
……
Adding 6 1/6's gives a result less than 1.
……
Adding 20 1/20's gives a result greater than 1.
……

[…@pc…]
```

#6, page 363. Filename: trap.c

```
[…@pc…]$ trap.out
Enter the number of subintervals = 2
Approximated area under the curve g = 3.875795
Approximated area under the curve h = 4.000000
[…@pc…] trap.out
Enter the number of subintervals = 128
Approximated area under the curve g = 5.869109
Approximated area under the curve h = 6.278594
```

#1, page 422. Filename: grade.c
This is a good program with which to practice top-down design. Your main() function may only contain user defined function calls, fopen(), fclose(), variable declarations, one loop, and the return statement. The only assignment statements allowed in main() are those for your two FILE*. I wrote six functions other than main(). main() called all six.

```
[…@pc…]$ cat examdat.txt
5 dbbac
111 dabac
102 dcbdc
251 dbbac
[…@pc…]$ grade.out
[…@pc…]$ cat report.txt
          Exam Report

Question  1  2  3  4  5
Answer    d  b  b  a  c

 ID   Score(%)
111     80
102     60
251    100

Question    1  2  3  4  5
Missed by   0  2  0  1  0
```

#12, page 428. Filename: binary.c
Note that to use the binary search , you need to first use selection sort function select_sort() to sort the elements in order. Your program will read from a file named **data.txt** that contains up to 20 elements. Once the elements are read into an array, your program should call select_sort(). Once the array is sorted, your program should the prompt the user for a value, and print out the position in the array of that value. If the value is not in the array, the program should indicate so. The program will terminate when the user enters -1.

```
[…@pc…]$ cat data.txt
12 7 8 17 2 20 19 16 14 3 2 1
[davis@lect2 p5]$ binary.out
Array before sort =  12  7  8 17  2 20  19  16  14  3  2  1
Array after sort  =  1  2  2  3  7  8  12  14  16  17  19  20

Please enter a value (-1 = done)> 3
3 is located at position 3 in the array.

Please enter a value (-1 = done)> 14
14 is located at position 7 in the array.

Please enter a value (-1 = done)> 4
4 is not in the array.

Please enter a value (-1 = done)> 90
90 is not in the array.

Please enter a value (-1 = done)> -1
[…@pc…]$
```