Project 7, Program Design

(100 points) A grocery store owner would like to know which fruits and vegetables are the top 5 of the month. The quantity of fruits and vegetables sold in the month is kept in a file. Write a program that reads in the data (a list of product in random order) from a file, and finds the top 5 in quantity.

Assume `fruits_vegetables.txt` has the format of name (one word string), unit price per pound (double), quantity (number of pounds, double), with each product on a separate line:

```
Grapefruit 1.99       31.3
Orange     1.49       102.5
Avocado    3.29       20.4
```

…

1. Name your program *`top5_products.c`*.
2. The program should be built around an array of `product` structures, with each `product` containing information of a fruit or vegetable's quantity, unit price, and name. Assume that there are no more than 100 items in the file.
3. Assume that name is one word, and no more than 100 characters.
4. Use fscanf and fprintf to read and write data.
5. Your program should include a sorting function so that it sorts the products in quantity. The function could use any sorting algorithms such as selection sort, insertion sort, etc.

```
void sort_products(struct product list[], int n);
```

6. Output the top 5 products in a text file called `top5.txt`, in the same format as the input file.

Before you submit:

1. Compile with –Wall. Be sure it compiles on ***student cluster*** with no errors and no warnings.

gcc –Wall top5_products.c

2. Test your program with the script. The script assumes the output file name is `top5.txt`.

chmod +x try_top5

./try_top5

3. Submit both top5_products.c and fruits_vegetables.txt (for grading purposes).

Total points: 100

1. A program that does not compile will result in a zero.
2. Runtime error and compilation warning 5%
3. Commenting and style 15%
4. Functionality 80% (functions were declared and implemented as required)

Programming Style Guidelines

The major purpose of programming style guidelines is to make programs easy to read and understand. Good programming style helps make it possible for a person knowledgeable in the application area to quickly read a program and understand how it works.

1. Your program should begin with a comment that briefly summarizes what it does. This comment should also include your **name**.
2. In most cases, a function should have a brief comment above its definition describing what it does. Other than that, comments should be written only *needed* in order for a reader to understand what is happening.
3. Information to include in the comment for a function: name of the function, purpose of the function, meaning of each parameter, description of return value (if any), description of side effects (if any, such as modifying external variables)
4. Variable names and function names should be sufficiently descriptive that a knowledgeable reader can easily understand what the variable means and what the function does. If this is not possible, comments should be added to make the meaning clear.
5. Use consistent indentation to emphasize block structure.
6. Full line comments inside function bodies should conform to the indentation of the code where they appear.
7. Macro definitions (#define) should be used for defining symbolic names for numeric constants. For example: `#define PI 3.141592`
8. Use names of moderate length for variables. Most names should be between 2 and 12 letters long.
9. Use underscores to make compound names easier to read: `tot_vol` or `total_volumn` is clearer than totalvolumn.