CPE 101
Winter 2009
Laboratory 7

Due Date

- **By the end of your last scheduled lab of week-eight.**
- **You must turn in your source electronically on vogon using the handin command – instructions are provided in below.**

Objectives

- To experience and understand records/structures in C.
- To learn about and create arrays of records.
- To practice using FILE I/O.
- To practice developing functions to sort array data.
- To develop a complete a C program, compile it, and turn it in electronically.

Resources

- Your instructor, peers, texts, and your own innate capabilities and resourcefulness!

Orientation…

This lab will involve developing a small C program which handles data about planets. You will be creating your own type, a record/structure, in order to store the planet information for our solar system. You will need to develop a variety of functions to process this data.

Part 0: Creating the data type and reading in data…

Recall, that in order to define your own data type you will need to specify:

```
typedef struct {
   /* variables associated with the structure */
} name_of_record;
```

Start out by defining a structure to store planet data.. Name this data type planetData. This structure must include:

A planet name (80 char maximum)
The planet's equatorial diameter (km.)
The planet's distance to the Earth (km.)
The number of natural satellites (i.e. moons) the planet has

You will also be writing a variety of functions, which operate on arrays of planetData. At this time, just develop:

A function to print out the data for each planet data in the array. (See below for examples).

Now write a main function which creates an array of planetData (Note that there are nine planets in our solar system**). Write the appropriate code to parse the planet data from an input file (named planet.data, found in the class examples directory) and store it in the array of planetData. Once the data has been stored, call your print routine in order to print out the data.

After reading the input file, the output of your print routine should look like this:

```
Planet 0:
        name: Mercury
        diameter: 2439.7 km.
        distance to the Earth: 2.219e+08 km.
        number of moons: 0
Planet 1:
        name: Venus
        diameter: 6051.8 km.
        distance to the Earth: 2.61e+08 km.
        number of moons: 0
Planet 2:
        name: Earth
        diameter: 6378.1 km.
        distance to the Earth: 0 km.
        number of moons: 1
Planet 3:
        name: Mars
        diameter: 3397 km.
        distance to the Earth: 4.013e+08 km.
        number of moons: 2
Planet 4:
        name: Jupiter
        diameter: 71492 km.
        distance to the Earth: 9.681e+08 km.
        number of moons: 63
Planet 5:
        name: Saturn
        diameter: 60268 km.
        distance to the Earth: 1.6585e+09 km.
        number of moons: 47
Planet 6:
        name: Uranus
        diameter: 25559 km.
        distance to the Earth: 3.1573e+09 km.
        number of moons: 27
Planet 7:
        name: Neptune
        diameter: 24764 km.
        distance to the Earth: 4.3059e+09 km.
        number of moons: 13
Planet 8:
        name: Pluto
        diameter: 119 km.
```

```
        distance to the Earth: 7.528e+09 km.
        number of moons: 1
```

.

**Note that recently Pluto was kicked out of the solar system (well it just no longer qualifies as a planet). Given that there is lots of old data (like our file) that includes Pluto, you should search through the input to test if Pluto appears. If it does, you should remove it from the array (and update any variables about the number of planets in the array).

Next, we will be adding functions in order to sort the array of data based on different data values in the planet data. We will be doing this by using the selection sort algorithm. Thus we will develop a functions to:

1) Find the planet which is closest to the Earth (within a given range of the array) and return its index location in the array

2) Swap two planetData elements in the array using their index values to determine which to swap.

3) Sort the array using the selection sort algorithm by using the above two functions. (i.e. repeatedly find the closest planet in the list and move it to the front of the list, while updating where the front of the list begins)

When writing the function to find the array element which is closest to the earth, you will need to iterate though the entire array and check the values of the distance variable for the smallest one. For your information only, below is an example of iterating through the planet data array and searching for a planet with a diameter within a certain range.

```
int CertainDiameter(planetData a[])
{
  int i;
  for (i=0; i < NUM_P; i++)
  {
     if (a[i].diameter > 3000 && a[i].diameter <
     5000 )
     {
       return i;
     }
  }
}
```

Once you have written these functions, include the necessary code to sort the planet data.

Once this is in place, you should be able to continue the ouput from above to include the following:

```
The planets sorted by distance to the earth are:
Planet 0:
```

```
        name: Earth
        diameter: 6378.1 km.
        distance to the Earth: 0 km.
        number of moons: 1
Planet 1:
        name: Mercury
        diameter: 2439.7 km.
        distance to the Earth: 2.219e+08 km.
        number of moons: 0
Planet 2:
        name: Venus
        diameter: 6051.8 km.
        distance to the Earth: 2.61e+08 km.
        number of moons: 0
Planet 3:
        name: Mars
        diameter: 3397 km.
        distance to the Earth: 4.013e+08 km.
        number of moons: 2
Planet 4:
        name: Jupiter
        diameter: 71492 km.
        distance to the Earth: 9.681e+08 km.
        number of moons: 63
Planet 5:
        name: Saturn
        diameter: 60268 km.
        distance to the Earth: 1.6585e+09 km.
        number of moons: 47
Planet 6:
        name: Uranus
        diameter: 25559 km.
        distance to the Earth: 3.1573e+09 km.
        number of moons: 27
Planet 7:
        name: Neptune
        diameter: 24764 km.
        distance to the Earth: 4.3059e+09 km.
        number of moons: 13
Planet 8:
        name: Pluto
        diameter: 119 km.
        distance to the Earth: 7.528e+09 km.
        number of moons: 1
```

(Optional) Once that is complete, note that it is relatively simple to create alternative sort criteria, for example with exactly the same code you've already written you can simply write one new function which finds the planet with the fewest moons (within a given range of the array) and change the call in your sort function to use this criteria and you can sort the data based on the number of moons and include output such as:

```
The planets sorted by number of moons are:
Planet 0:
        name: Mercury
```

```
        diameter: 2439.7 km.
        distance to the Earth: 2.219e+08 km.
        number of moons: 0
Planet 1:
        name: Venus
        diameter: 6051.8 km.
        distance to the Earth: 2.61e+08 km.
        number of moons: 0
Planet 2:
        name: Earth
        diameter: 6378.1 km.
        distance to the Earth: 0 km.
        number of moons: 1
Planet 3:
        name: Pluto
        diameter: 119 km.
        distance to the Earth: 7.528e+09 km.
        number of moons: 1
Planet 4:
        name: Mars
        diameter: 3397 km.
        distance to the Earth: 4.013e+08 km.
        number of moons: 2
Planet 5:
        name: Neptune
        diameter: 24764 km.
        distance to the Earth: 4.3059e+09 km.
        number of moons: 13
Planet 6:
        name: Uranus
        diameter: 25559 km.
        distance to the Earth: 3.1573e+09 km.
        number of moons: 27
Planet 7:
        name: Saturn
        diameter: 60268 km.
        distance to the Earth: 1.6585e+09 km.
        number of moons: 47
Planet 8:
        name: Jupiter
        diameter: 71492 km.
        distance to the Earth: 9.681e+08 km.
        number of moons: 63
```

Part 3: Handing in Your Source Electronically…

1. Log on to vogon using the Secure Shell Client program (or your favorite equivalent).

2. Change directory (cd-command) to the directory containing the source file or files to hand in.

3. Execute the following command –

```
handin zwood csc101lab07 lab07.c
```

4. You should see messages that indicate handin occurred without error. You can (and should) always verify what has been handed in by executing the following command:

```
handin zwood csc101lab07
```