AOSC 652: Analysis Methods in AOSC

Assignment #4a: Data Sorting, Input/Output, Subroutine Calls

Due: Thursday, 22 Sept 2016 (noon)

Late penalty: 10 pts per day

Name: ____

1. (40 points) This assignment provides further practice in FORTRAN, applied to various algorithms for sorting integers. Please read pages 320 to 329 of *Press et al., Numerical Recipes* as part of this assignment.

In subdirectory ~rjs/aosc652/week_04, you will find a series of files named random*integer*dat, that contain randomly sorted integers ranging from 1 to 10, 1 to 1000, etc. The largest file contains a random list of integers from 1 to 1,000,000. The filenames are self explanatory.

In class on Monday, we worked with computer program:

~rjs/aosc652/week_04/sort_numbers.f

We started with a subroutine that sorts the input integer based on the **piksrt** algorithm, as given on page 321 of Press *et al.*, and altered this subroutine (based on manner in which it is given in *Press et al.*) *so that it conforms to our preference for retaining the values of input and output arrays*. The associated files are piksrt.f and sort_numbers.class.f in ~rjs/aosc652/week_04.

Your assignment is to:

a) alter the subroutine for sorting integers based on the **heapsort** method, described on pg 327–329 of Press *et al.* and given in file ~rjs/aosc652/week_04/heapsort.f, so that it *conforms to our preference for retaining the values of input and output arrays*.

> **Please note the starting code for heapsort is identical to the code in Press *et al.* except we have replaced the variable written as the letter "l" with the letter "k". Since the letter "l" and the number "1" are so difficult to distinguish, we try to avoid every using the letter "l" in our codes ☺ (capital L is used as it is easy to distinguish).**

b) compile sort_numbers.f including the new **heapsort** subroutine

c) assess the timing performance of:

> **piksrt** for sorts of the files containing 10000, 100000 integers
>
> **heapsort** for sorts of files containing 10000, 100000, 300000, and 10000000 integers.

A nice aspect of this exercise is that you can tell by simple inspection whether the sort routine has worked ☺

You can compute the time it takes for your code to execute by going into the **s-shell**, which is done by typing **sh** into the linux command line, and then running the following commands:

```
time echo random_integers_1_10000.dat | sort_numbers.e
time echo random_integers_1_100000.dat | sort_numbers.e
time echo random_integers_1_300000.dat | sort_numbers.e
time echo random_integers_1_1000000.dat | sort_numbers.e
```

where the command **sort_numbers.e** should run the code you have created. Enter **exit** when done.

Note: it may be best to work with a single FORTRAN program that contains two subroutines for sorting data, **piksrt** and **heapsort**. You can choose which one is selected by simply *commenting out* the call to the routine that will not be used (this is an inelegant method but one that will meet our needs). If you choose this option, you will have to recompile the code when switching from **piksrt** to **heapsort**.

To understand the dependence of run time on the number of variables being sorted *and* the method used to sort the variables, record below the numerical values of run time (seconds) for the various sorts. This information can be obtained from the line "**user**" (middle output line) given by the **time** command described above.

Number of Integers	**piksrt** Runtime (sec)	**heapsort** Runtime (sec)
10,000		
100,000		
300,000	-------	
1,000,000	-------	

Please turn in the table given above, your code (printed using enscript and the full path name of the file), and answers to the following questions:

What is the simple mathematical expression that describes how the runtime of **piksrt** varies with respect to the number of integers being sorted?

Based on your answer to the question given above and the numerical info in Table 1, estimate how long (in minutes) **piksrt** would take to sort 300,000 and 1 million integers?

What is the simple mathematical expression that describes how the runtime of **heapsort** varies with respect to the number of integers being sorted?

If we need to sort through 1 million data points, as could for example be the case if we are looking at ground level ozone measurements obtained in the Eastern U.S. by the EPA CASTNET network (http://www.epa.gov/castnet/) over the past ~20 years, how much faster (i.e., ratio of speeds) could **heapsort** sort these data compared to **piksrt**?