Assignment 7

Submit solutions to 2 and 3. For 2, submit the plots that you produce.

1. Modify the `selection.c` program (available on the Moodle site) so that it finds and displays the kth largest value in an integer array that is divisible by 3, where k is specified by the user.
   Sample **input** and **output**:

   ```
   Enter the value for k: 2
   1 15 2 16 3 17 4 18 5 19
   The 2th largest number divisible by 3 is 15
   ```

   The program should write an appropriate message in the case that there are less than k integers in the list divisable by 3

2. Type the code for the `QuickSelect` algorithm given in the notes into an editor and save it to a file called `quickselect.c` Now carry out the following exercise. Note that it is sufficient to carry out the following exercise, modifying the size of the array, as discussed below, by hand and recompiling the code each time. For an extra challenge, consider modifying the code so that it generates the required table automatically, by looping through a set of different array sizes.

   a. Download the file `qselect_funcs.c` from the moodle web-site and put it in the same folder as the `quickselect.c`. Then make the following modifications to the quick select program:

   b. Add two new double precision variables, `start` and `end` in the declaration section of the program.

   c. Just before the `while` loop, add the following lines to the code:

   ```
   start = get_cpu_time();
   srand(time(0));
   ```

   d. Just after the while loop, before the answer is printed, add the following lines to the code:

   ```
   end = get_cpu_time();
   printf("The program took %lf seconds\n", (end-
   start));
   ```

   e. Add an include statement at the top of the file to include the `time.h` header file. After the `#include` and `#define` statements, put in the following two statements:

   ```
   double get_cpu_time();
   void read_data_from_file(char fname[], double
   a[], int size);
   ```

   The `get_cpu_time()` function is a special function to calculate the run-time of a program. It is implemented in the file

qselect_funcs.c (see below about how to compile). The `srand()` function is a function to "seed" the random number generator. It ensures that, each time that you run the program, it generates a different set of random numbers.

f. To read values from the input file, replace the following lines

```
for (i = 0;i<MAX_SIZE;i++)
    {
      scanf("%lf", &a[i]);
    }
```

in the original listing with the line

```
read_data_from_file("array.dat", a, MAX_SIZE);
```

g. To compile the program **on a windows machine** do the following (all on a single line)

```
gcc –D_WINDOWS –wl,--stack,100000000 –o qselect.exe quickselect.c
qselect_funcs.c
```

Otherwise (on a mac or other), the following compilation is sufficient:

```
gcc –o qselect.exe quickselect.c qselect_funcs.c
```

h. Run the program by typing

```
qselect.exe
```

and note the size of the array `MAX_SIZE` and the run-time of the program. Note that each time that you run the program on the same input, the program has a slightly different run-time.

i. Repeat f. and g. for a number of different sizes of the array to create a table (select a range of sizes that allow you to see the performance of the algorithm e.g. as below)

Size	Average Run Time (over 5 runs)
100	0.000025
500	…
1000	…
5000	…
10000	…

j. Carry out the same set of steps for the simple selection algorithm (`select.exe` from the web-site), to again generate a table of sizes and average run-times.

k. Using Excel or any other application for plotting data, plot the run-time against size of the two algorithms.

l. Can you convince yourself, that the QuickSelect algorithm is linear and the Select algorithm is quadratic?

3. Modify the `QuickSelect` algorithm so that it outputs the kth largest number less than a given threshold value, T, which is read in from the user.