ENGR 121: Computation Lab I

Handout for Lab 7: Basic Statistics, Searching, and Sorting

Practice Exercises

1. Write a function that will return the mean of the values in a vector, not including the minimum and maximum values. Assume that the values in the vector are unique. It is okay to use the built-in **mean** function. To test this, create a vector of 10 random integers, each in the range from 0 to 50, and pass this vector to the function.

2. A moving average of a data set $\mathbf{x} = x_1, x_2, x_3, x_4, \ldots, x_n$ is defined as a set of averages of subsets of the original data set. For example, a moving average of every two terms would be $\frac{1}{2}(x_1 + x_2, x_2 + x_3, x_3 + x_4, \ldots, x_{n-1} + x_n)$. Write a function that will receive a vector as an input argument, and will calculate and return the moving average of every two elements.

3. A *median filter* on a vector has a size; for example, a size of $n = 3$ means calculating the median of every three values in the vector. The first and last elements are left alone. Starting from the second element to the next-to-last element, every element of a vector *vec(i)* is replaced by the median of $[vec(i-1), vec(i), vec(i+1)]$. For example, if the signal vector is

```
signal = [5 11 4 2 6 8 5 9],
```

applying a median filter with $n = 3$ yields the new signal

```
medianFilter3 = [5 5 4 4 6 6 8 9]
```

Write a function to receive the original signal vector and return the median filtered vector. The size n of the filter must be passed as an input argument.

4. A function *generatevec* generates a vector of n random integers (where n is a positive integer), each in the range from 1 to 100, but, all of the numbers in the vector must be different from each other (no repeats). So, it uses **rand** to generate the vector and then uses another function *alldiff* that will return logical 1 for true if all of the numbers in the vector are different, or logical 0 for false if not in order to check. The *generatevec* function keeps looping until it does generate a vector with n non-repeating integers. It also counts how many times it has to generate a vector until one is generated with n non-repeating integers and returns the vector and the count. Write the *alldiff* function.

5. Write a function that will receive a vector as an input argument, and will print all of the values from lowest to highest in the vector, until the mean of the numbers is reached (including the mean). For example, if the input vector has the values $[5, 8, 2, 4, 6]$, the mean is 5 so the function would print $[2, 4, 5]$.

6. Write a function *mydsort* that sorts a vector in descending order (using a loop, not the built-in sort function).

7. Write a function *mymedian* that will receive a vector as an input argument, and will sort the vector and return the median. Any built-in functions may be used, except the median function. Loops may not be used.

8. In statistical analyses, quartiles are points that divide an ordered data set into four groups. The second quartile, $Q2$, is the median of the data set. It cuts the data set in half. The first quartile, $Q1$, cuts the lower half of the data set in half. $Q3$ cuts the upper half of the data set in half. The interquartile range is defined as $Q3 - Q1$. Write a function that will receive a data set as a vector and will return the interquartile range.

9. Write a function that will receive a vector and will return two index vectors: one for ascending order and one for descending order. Check the function by writing a script that will call the function and then use the index vectors to print the original vector in ascending and descending order.