CSC 362 Programming Assignments #4 & 5
Due Date: Monday, November 14

These two assignments require that you implement two Intel assembly language programs embedded in C programs. If you are using Windows, you will have to use Visual Studio. If you are using a Mac, you will have to use Xcode. Otherwise, you will have to log in to the NKU virtual lab to access a virtual Windows 10 computer running Visual Studio.

Program 4: An increasingly additive sequence is a list of integer numbers such that the next number in the sequence is greater than the sum of the previous numbers. For instance, 1, 2, 4, 9, 21, 50 is increasingly additive because 2 > 1, 4 > 2 + 1, 9 > 4 + 2 + 1, 21 > 9 + 4 + 2 + 1 and 50 > 21 + 9 + 4 + 2 + 1. We use increasingly additive sequences for private key encryption. An increasingly additive prime number sequence is the same but each number must be a prime number as well. So for instance, 2, 5, 11, 19, 41 is such a sequence while 2, 5, 9, 19, 41 is not (9 is not a prime) and 2, 5, 11, 13, 41 is not because 2 + 5 + 11 > 13. Write a program which, in C, declares any necessary variables and inputs two numbers (assume the first number will be less than the second) and generates the sequence of all numbers that form an increasingly additive prime sequence between them. For instance, if input with 4 and 20 the sequence is 5, 7, 13. If input with 10 and 100, the sequence is 11, 13, 29, 59. Your program should determine each number in the sequence and print it out before going on to the next number. You should implement this program in C before trying to convert it to assembly. Test your program on the above example inputs (4, 20 and 10, 100) and then run it on 3, 1000 and 300, 5000.

Program 5: Implement the Selection Sort algorithm for an array of int values in assembly language. The selection sort makes n-1 passes through an array of n elements locating the smallest element and swapping it with the ith element. For instance, on the first pass, i=0, locate the smallest element in the range of j=0 to j=n-1 and swap it with array[0]. On the second pass, i=1, locate the smallest element in the range from j=1 to n-1 and swap it with array[1]. The algorithm is shown below.

```
for(i=0;i<n-1;i++) {
      min = a[i];
      minposition = i;
      for(j=i+1;j<n;j++) {
            if(a[j] < a[minposition]) {
                  min = a[j];
                  minposition = j;
            }
      swap a[i] and a[minposition]
      }
}
```

Implement your code in assembly language with the array declared and initialized (or input) in C code and after the assembly code, output the sorted array in C code. Note that you may find this program tricky because you are limited to 4 data registers. Use only one array (do not manipulate a copy of the array). Run the program on the 2 data sets below.

Run #1: 100, 99, 97, 95, 90, 87, 86, 83, 81, 77, 74, 69, 63, 50, 44, 43, 39, 31, 29, 12
Run #2: 8, -1, 7, -8, 6, 3, -4, 4, 1, -2, 9, -5, 0, 2

For both programs, submit the source code and output. In both cases, the output can be added to your source code in comments at the bottom. Make sure your code is well-commented.