Assignment 02

Instructor: Mehrdad Nojoumian

Course: Data Structure and Algorithm Analysis

Deadline: July 05

Review lectures on C++, C++ examples and "More On Algorithms", then write a program in C++ or the language of your choice for the following algorithms. You should write your code to match the pseudocode for each algorithm. Code for each algorithm should be in a separate file. Do not use any headers or libraries that are not included in standard installations of whichever language you chose. Code that does not compile will receive at most half credit.

1. Write code implementing Selection Sort. Your program should let the user enter how large of an array they wish to sort, followed by the values they wish to be in the array. It should print the unsorted and sorted array. Start with the bubblesort example from class/notes and replace the bubblesort function with a select sort function.

```
proceedure SelectionSort(array a, length(A) = n)
        for i in 0 to n - 2
                maxIndex = i
                for j in (i + 1) to (n - 1)
                        if a[j] > A[maxIndex]
                                maxIndex = j
                tmp = A[i]
                A[i] = A[maxIndex]
                A[maxIndex] = tmp
```

2. Extended Euclidian Algorithm (page 4 of last lecture, more on algorithms) Write code that asks for and gets two integers, then computes and displays their greatest common divisor using the Extended Euclidian Algorithm (EEA). The EEA should be implemented as a function that takes two integers as arguments and prints their GCD. Define 3 temp values to fix the following code if it doesn't work!

```
proceedure EEA(int: a, b)
        s = 0
        old_s = 1
        t = 1
        old_t = 0
        r = b
        old_r = a
        while r != 0
                q = old_r div r
                (old_r , r) ← (r , old_r - q * r)
                (old_s , s) ← (s , old_s - q * s)
                (old_t , t) ← (t , old_t -  q * t)
        print("GCD=" old_r)
        return (t,s)
```

3. Finding 2 largest numbers in a list (page 5 of last lecture, more on algorithms)

TwoLargest should be written as a function that is called in main. The user should be able to enter any list of numbers that they wish. **Hint**: You can use parts of the code from the bubblesort example such as vector_get() to create the list of numbers. Additionally, length(A) = n can also be found with A.size() in C++ so TwoLargest can be written with only one argument.

```
proceedure TwoLargest(A = [a1...an], length(A) = n)
        large_1 = 0
        large_2 = 0
        for i = 1 to n
                if A[i] > large_1
                        large_2 = large_1
                        large_1 = A[i]
                else if large_2 < A[i]
                        large2 = A[i]
```

4. Write a program where you implement Modular Exponentiation (ME) using the square and multiply approach as a function which is called in main. ME calculates a^k mod n. The program should get values for a, k and n from the user. This code requires two steps. First k must be converted to a binary representation K consisting of a list of 0s and 1s. Second, Modular Exponentiation must be performed using a, n and K[] as arguments.

```
procedure BinaryK(k)
        K = empty list //hint: make K a vector
        tmp = k
        i = 0
        while tmp > 0
                add tmp mod 2 to K //hint: use pushback
                tmp = (tmp-K[i])/2
                i++
        return K

procedure ModularExpo(a, K, n)
        if n = 1
                return 0
        b = 1
        if K = 0
                return b
        A = a
        if K[0] = 1
                b = a
        for i = 1 to length(K)-1
                A = A*A mod n
                if K[i] = 1
                        b = A*b mod n
        return b
```