# Homework 1

As part of this assignment, you will practice your understanding of C.

# Problem 1

You are given a number N, where N is a natural number such that N >=1. Write a program that prints the first Fibonacci numbers. The program will accept a single command-line argument which is N. The output will include the first N Fibonacci numbers, each printed on a new line. As an example, calling ./a.out 10 should print:

0  
1  
1  
2  
3  
5  
8  
13  
21  
34  
55

# Problem 2

You are given a number N, where N is a natural number such that N >=1. Write a program that prints all the prime numbers between 1 and N (inclusive). The program will accept a single command-line argument which is N. The output will include on the first line the number N, followed by a prime on each of the following lines. As an example, calling ./a.out 10 should print:

10  
1  
2  
3  
5  
7

# Problem 3

You will have to implement bubble sort to sort a doubly-linked list of integers in ascending order. As a starting point, I’m providing problem2\_template.c. The template provides build\_list, which takes as an argument the length of the list to be generated. The function generates a sequence of random numbers that are to be sorted. The print\_list function is a utility that prints the list. You have to implement two approaches to perform bubble sort.

* In bubble\_sort\_copy\_value, you have to implement bubble sort by manipulating the value field in the linked list. So, you can perform the bubble sort by swapping the values of two consecutive elements. This function should be easier to implement since you do not have to reorganize the list.
* In the bubble\_sort\_copy\_ref, you must implement bubble sort without modifying the value fields (you can read the values and do not write to them). The function has to be implemented by manipulating the pointers of the elements in the linked list. This will require some careful consideration of how pointers must be assigned.

Output: The output of the code should be the sorted array.

Hint: For the bubble\_sort\_copy\_ref, I found the following illustration useful. Originally, we need to consider four variables before\_a, a, b, after\_b that refer to elements in the list. I’m allowing for prev\_a and next\_b to be pontetially NULL (which happens in the beginning/end of the list). The next and prev pointers for these are as follows:

before\_a ===(next)==> a ===(next)==> b ===(next)==> after\_b  
 before\_a <==(prev)=== a <==(prev)=== b <==(prev)=== after\_b

If we need to switch the values of a and b, we need to switch the pointer around so that we end up with the following organization:

before\_a ===(next)==> b ===(next)==> a ===(next)==> after\_b  
 before\_a <==(prev)=== b <==(prev)=== a <==(prev)=== after\_b

Now, you need to think about changing the pointers of before\_a, a, b, and after\_b according to the illustrations shown above. Be careful about when before\_a and after\_b are NULL. If the list has at least two elements, a and b cannot be NULL.

Drawing the relationships between variables as you step through code is quite useful to debug. Remember gdb (or CLion’s debugger) and Valgrind are your friends. Use them in times of trouble!

# Problem 4

As part of this problem, you will write a program that has as input a CSV file and outputs another CSV file. The program will be invoked with the names of the input and output files as arguments. The input file has two columns that include positive integers. The program will output a file with the same two columns as the input CSV. In addition, the last column is the sum of the first two columns.

Example: Consider the input to the program is a file input.csv that includes the following content:

0,9  
1,2  
4,5  
11,434  
3232,999

The output will be: 0,9,9  
1,2,3  
4,5,9  
11,434,445  
3232,999,4231