IC250 Lab 1

This assignment has two questions.

1. Write a program that inserts a given integer into the correct position in a sorted array of integers. The input array is to be read from a file. The output, which will contain one more integer, has to be written into a new file.

   You may assume that the input file is already sorted, and the size of the input is provided as an input.

   The input file, its size, the new integer, and the output file have to be specified as command line arguments. The program must use a function having the following prototype (or something similar):

   ```
   int* insert(int array, int len, int new);
   ```

   where `array` is the input array read from file, `len` is the length of this array, and `new` is the new integer to be inserted. The function returns back an array of `len+1`, with `new` inserted into the proper position. The insertion into the input array has to be done within the function. `len` and `new` are provided as inputs via command line arguments.

   Your main program may look like this:

   ```
   int main(int argc, char* argv) {

     /* read cmd line args, open input file etc. */

     newarray = insert(array, len, new);

     /* write out the new array into the output file */

   }
   ```

   A sample run is shown below:

   ```
    $ cat input.txt <ENTER>
    45
    56
    78
   ```

```
89
102
$ a.out input.txt 5 67 output.txt <ENTER>
$ cat output.txt <ENTER>
45
56
67
78
89
102
```

Optional:

(a) You need not assume that the input file is sorted, nor its size is known.

2. Write a program called `countArgs` which determines the frequency of words passed as command line arguments. The behaviour of the program is illustrated below:

```
$ countArgs hello world ins hello ins fortune world world <ENTER>
hello 2
world 3
ins 2
fortune 1
```

You may assume that the maximum number of ***distinct*** *input words will at most be 10.* This is *not* the total number of input words, which could be greater than 10.

One way of doing this is to create a structure like this:

```
struct data {
   char* word;
   int count;
}
```

Create an array of structures, and use an element of this array to represent each word. Initially the count of all will be zero, and it gets incremented each time a word appears in the command line arguments.

Optional:

(a) You can check that the maximum number of unique input words is at most 10, else show an error.

(b) You can remove the restriction that the maximum number of unique input words is 10.