

## Question 2: Recursion

### 2a. Recursive functions

- Implement the following functions using recursion (*see the next page for examples*)
  - Count up from 0 to n. (*print to stdout*)
  - Count down from  $2n$  to 0 by 2 (*print to stdout*)
  - nth, nth\_sorted
    - this applies to Sorted Lists from Q1a
  - remove\_nth, remove\_nth\_sorted
    - this applies to Sorted Lists from Q1a

*note: no marks will be awarded if implemented iteratively,  
even if the answer produced is correct when run*

### 2b. Greatest Common Divisor (GCD)

- Read up on the Euclidean Algorithm for solving GCD in Wikipedia
  - First Section: [https://en.wikipedia.org/wiki/Euclidean\\_algorithm](https://en.wikipedia.org/wiki/Euclidean_algorithm)
  - Procedure: [https://en.wikipedia.org/wiki/Euclidean\\_algorithm#Procedure](https://en.wikipedia.org/wiki/Euclidean_algorithm#Procedure)
  - Example: [https://en.wikipedia.org/wiki/Euclidean\\_algorithm#Worked\\_example](https://en.wikipedia.org/wiki/Euclidean_algorithm#Worked_example)
  - Using mod: [https://en.wikipedia.org/wiki/Euclidean\\_algorithm#Euclidean\\_division](https://en.wikipedia.org/wiki/Euclidean_algorithm#Euclidean_division)
  - Implementations: [https://en.wikipedia.org/wiki/Euclidean\\_algorithm#Implementations](https://en.wikipedia.org/wiki/Euclidean_algorithm#Implementations)
- Examine the final implementation, which is recursive, and implement it in C
  - it should have the signature `long long gcd(long long, long long)`
- This implementation (as yours should be) is naturally “tail recursive”
  - **explain why it is tail recursive in the readme**
  - **make sure your make file uses the appropriate gcc flag** to run tail recursive code efficiently

### To test question 2

Write a program called `a4q2.c`

- The program must read in a text file that contains a series of commands, one per line, with the name of the text file entered as a command line argument
  - Base this code on the code you used in q1a to implement command entries from a file
  - However, the code will need to be extended to allow for new commands that are detailed below
- Some of the questions in 2a use the Sorted\_List data type with key\_type as double and value\_type as int, the same as `a4q1a_int.c`.
  - So when compiling files containing Sorted\_List functions in your make file, use `-DINT`
- All new commands for entry from the input file are listed on the next page

The assignment concludes with Question 3: Fraction ADT  
to be released early next week

## List of Commands from the Input File

All commands from q1a should be made available as well as the following

- `count_up n`
  - Prints the integers from 0 to  $n$  on a single line, comma separated, with 5 spaces before
  - Using the following commands  
`count_up 4`  
the output should be  
`count_up from 0 to 4`  
`0, 1, 2, 3, 4`
- `count_down n`
  - Same as `count_up`, but printing the integers down from  $2n$  to 0 on by 2
  - Using the following commands  
`count_down 4`  
the output should be  
`count_down from 8 to 0 by 2`  
`8, 6, 4, 2, 0`
- `nth n order`
  - Displays the  $n$ th element in the list according to the order specified (inserted or key sort) on its own line as *key value*
    - Indented 5 spaces with 2 spaces between the key and the value
    - Firsts prints the command – see below for an example
  - Using the input from the append examples in q1a and the following commands  
`print_all`  
`nth 1 INSERTED_ORDER`  
`nth 1 SORTED_ORDER`  
`nth 5 INSERTED_ORDER`  
the output should be  
`print_all: Insertion Order`  
`3.27 1427`  
`0.94 984`  
`7.21 346`  
`nth: n = 1, Insertion Order`  
`0.94 984`  
`nth: n = 1, Key Sort Order`  
`3.27 1427`  
`nth: n = 5, FAILED, n >= size where size = 3`
- `remove_nth n order`
  - removes the  $n$ th element in the list according to the order specified (inserted or key sort) and displays the removed element on its own line as *key value*
  - Again using the input from the append examples in q1a  
For the commands  
`remove_nth 2 INSERTED_ORDER`  
`print_all`  
`remove_nth 1 SORTED_ORDER`  
`print_all`  
the output should be  
`remove_nth: n = 2, Insertion Order`  
`7.21 346`  
`print_all: Insertion Order`  
`3.27 1427`  
`0.94 984`  
`remove_nth: n = 1, Key Sort Order`  
`3.27 1427`  
`print_all: Insertion Order`  
`0.94 984`

- `gcd x y`
  - displays the greatest common divisor of long long numbers `x` and `y`
  - Using the following commands  
`gcd 24 18`  
the output should be  
`gcd(24, 18) == 6`

*Note: This command can be left out provided you have demonstrated that `gcd` behaves properly by having successfully used it in the Fraction section of q3*