- **These must be completed and shown to your lab TA either by the end of this lab, or by the start of your next lab.**
- **You are required to work with a partner for this lab.**

The following questions and exercises are designed to heighten your understanding of asymptotic analysis and loop invariants.

1. While finding an asymptotic bound for `n(nlgn + 1)`, is it justifiable to simplify it to the functions on the right? Explain why or why not in a few short sentences.

   - `n(nlgn + 1) => n(nlgn)`
   - `n(nlgn + 1) => n(n + 1)`
   - `n(nlgn + 1) => nlgn + 1`

2. Is O(n) a good bound on cn/lgn? Why or why not?

3. Consider the following function:

```
int fact(int x) {
  if (x<1) return 1;
  return x * fact(x-1);
}
```

   Rewrite `fact` to be tail-recursive. Hint: use a helper function.

   Rewrite `fact` iteratively. What is the loop invariant?

4. Open the provided file `queuestack.cc` and compile it using the command `make queuestack`. You should recognize the code for the QueueStack class from the midterm. Your task is to write an efficient implementation of `dequeue_mult` and `dequeue_mult_back`:

```
// TODO
// parameters: number of data elements to return
// returns array containing (num) data elements from the front of the QueueStack
int* dequeue_mult(int num) {
   return nullptr;
}

// TODO
// parameters: number of data elements to return
// returns array containing (num) data elements from the back of the QueueStack
int* dequeue_mult_back(int num) {
   return nullptr;
}
```

What are the runtimes of your implementations as a function of a) num, and b) QueueStack size? Why is it efficient? Justify your answers.

How would the implementation and performance of `dequeue_mult_back` change if the `Node` structures had no `prev_` pointers?