Computer Programming: Skills and Concepts
Tutorial 3
Week 10 — November 22-25, 2010

Programming

1. The $\log^*$ function (different from log itself - the * is important), which is applied to a number n, is defined to be the number of times one has to repeatedly apply log to n before the answer becomes less than or equal to 1. For example, assuming base 10 logs, we have that:
$\log^*(5) = 1$, $\log^*(50) = 2$, $\log^*(500) = 2$, $\log^*(5000000) = 2$, $\log^*(10^{50}) = 3$.

Assuming that double log(double x) is the log function, write a *recursive* `logstar` to compute $\log^*$. The type of this function will be:

```
int logstar(double x)
```

2. Now write a new function `logstar2` which computes the same answer but using *iteration* (ie, a `for` or a `while`) rather than recursion. The type of the function is identical:

```
int logstar2(double x)
```

Fibonnaci

In class in lecture 25 we saw a recursive algorithm which computes the value of F(n) the n-th Fibonnaci number. We also showed on the board, how we could compute F(n) *iteratively* (via a loop, not recursively).

Write an *iterative* program (ie use `for` or `while`) which computes *all* the fibonnaci numbers for 1 to 20 and stores them in a global array `fibs`. The type of the function should be:

```
int twentyfibs(void)
```

Testing

Make a new copy of `mergesort.c` called `mergerec.c` and replace the original code of the `mergesort` function with the new recursive code on slide 14 of Lecture 25. Do some run-time experiments with this new recursive function, and compare to the results for original mergesort.