Problem 1-1

For each algorithm we know that it takes f(n) = y microseconts (i.e for 10 inputs, lg 10 takes 1 microsecond). So f(n) = y. To figure out how many inputs the algorithms can solve, we do the following:

We inverse the function and then replace y with $10^6, 6 * 10^7, 36 * 10^8, 864 * 10^8, 2592 * 10^9, 31536 * 10^{10}, 31536 * 10^{12}$ respectively.

So we have the following inverses:

- For $\lg n = y$, we get $10^y = n$.
- For $\sqrt{n} = y$, we get $y^2 = n$.
- For $n = y$, we get $y = n$.
- For $n * \lg n = y$, we cannot get inverse. We will use Newton's method for this (Numerical Analysis), in a Python script.
- For $n^2 = y$, we get $\sqrt{y} = n$.
- For $n^3 = y$, we get $\sqrt[3]{y} = n$.
- For $2^n = y$, we get $\log y = n$.
- For $n! = y$, we will have to divide by consecutive numbers, starting from 1 to find n.

Convention: A month has 30 days and a year 365.

The Python code we used is the following (can also be found in newton_method.py):

```
import math

N = int(input("Equals to: "))

def create_estimate (x):
    "We estimate as x/logx to begin the search for a root."
    return x/math.log(x,10)

def f (x):
    "x*lgx function"
    return x * math.log(x, 10)
```

```
def f_der (x):
    "lgx + 1/ln10"
    return math.log(x,10) + 1/math.log(10)

estimate = create_estimate(N)
print ("Estimate is ", estimate)
new_estimate = estimate - ((f(estimate) - N)/f_der(estimate))


while (abs(new_estimate - estimate) > 0.1 ):
    estimate = new_estimate
    new_estimate = estimate = estimate - ((f(estimate) - N)/f_der(estimate))

print (new_estimate)
```

This is the final result:

	1 sec.	1 min.	1 hr.	1 day.	1 mon.	1 yr.	1 cent.
$\lg n$	10^{10^6}	10^{6*10^7}	10^{36*10^8}	10^{864*10^8}	$10^{2592*10^9}$	$10^{31536*10^{10}}$	$10^{31536*10^{12}}$
$\sqrt{n}$	10^{12}	$36 * 10^{14}$	$1296 * 10^{16}$	$746496 * 10^{16}$	$6718464 * 10^{18}$	$994519296 * 10^{20}$	$994519296 * 10^{24}$
n	10^6	$6 * 10^7$	$36 * 10^8$	$864 * 10^8$	$2592 * 10^9$	$31536 * 10^{10}$	$31536 * 10^{12}$
$n * \lg n$	189481	8649296	417596733	8692884131	228202745102	23582565190405	2059329031596633
n^2	1000	7746	60000	293940	$161 * 10^3$	$17758 * 10^3$	$17758 * 10^4$
n^3	100	391	1532	4420	13737	68067	315940
2^n	19	25	31	36	41	48	54
$n!$	9	11	12	13	15	16	18