Q1.

5) input: positive integer n; Output: $x=2^{2^n}$

```
x = 2;
i = 0;
while (i < n)
 {
  x = x * x;
  i = i +1;
 }
```

Running Time of this algorithm is $O(n)$ assuming multiplication takes $O(1)$ time.

Let multiplications take O(n^2) time for two n-bit numbers.

Notice at that step i we multiply 2^2^i with itself. Each of these numbers takes $O(2^i)$ bits to store. Therefore the cost of each multiplication is $O(2^{2i})$

Total time is: sum_{i=1}^n $2^{2i} = 2^{2n} + 2^{2(n-1)} + 2^{2(n-2)} + \dots + 2^{2(n-(n-1))}$

$= 2^{2n}(1 + 1/4 + 1/16 + \dots + 1/2^{2(n-1)})$

$\le 2^{2n} \times 2 = O(2^{2n})$; also $O(4^n)$

Another solution is:

a = 1 << n

v = 1 << a

The first line computes 2^n via bit-shifts, and the second 2^{2^n} via bit-shifts too.

The second line dominates, since we are doing 2^n bit shifts.

The complexity is $O(2^n)$

Q2.

1.31. (a) By the equation $\log N! = \log 1 + \log 2 + \cdots + \log N$ we can easily see that $N!$ is approximately $\Theta(N \cdot \log N) = \Theta(N \cdot n)$ bits long.

(b)We can compute $N!$ naively as follows:

<u>factorial</u> (N)
$f = 1$
for $i = 2$ **to** N
$f = f \cdot i$

We have N steps. In each step we multiply a n bit number by a nN bit number in the worst case. Therefore, total time is $N^2n^2 = N^2 (\log N)^2$

Q3.

The gcd of 1492 and 1776 via factorization is:
1492 = 2 x 2 x 373
1776 = 2 x 2 x 2 x 2 x 3 x 37
Common factors: 2 x 2, therefore 4 is the gcd

Now using Elucid's method, we have:
(1776, 1492) with 1776 = 1 x 1492 + 284
(1492, 284) with 1492 = 5 x 284 + 72
(284, 72) with 284 = 3 x 72 + 68
(72, 68) with 72 = 1 x 68 + 4
(68, 4) with 68 = 17 x 4 + 0
(4,0)
Gcd = 4

Now for the linear combination, we have:
4 = 1 x 72 – 1 x 68
= 1 x 72 – 1 x (284 – 3 x 72)
= -1 x 284 + 4 x 72
= -1 x 284 + 4 (1492 – 5 x 284)
= 4 x 1492 - 21 x 284
= 4 x 1492 – 21 (1776 – 1492)
= -21 x 1776 + 25 x 1492