1:

a = MAX(r, l*b)

n^r+ n^l^b = n^r+n^(l*b), so we have O(n^(MAX(r, l*b)))

2:

We will show that this is not possible by showing that there is a polynomial time reduction from a problem in P to a known EXPTIME-complete problem, and then concluding that if the reduction worked both ways, then P would have to be equal to EXPTIME, which is known to be false.

AFSOC that the claim held.

Let B the problem of determining whether a deterministic Turing machine completes in k steps for a TM with a non-unary alphabet, and outputting its result if it does complete. This is a problem known to be EXPTIME-Complete, but we could also intuitively justify this by realizing that k can be represented in log(k) bits, meaning that we would have to simulate the TM for a number of steps exponential in the input.

Let A be the Horn-Satisfiability problem from last week, which from OH we know to be P-Complete. We know that we have an algorithm to solve Horn-Satisfiability problem, from some known c, in O(n^c).

We see that we can naively construct a polynomial time reduction from B to A by running a TM for k=2^(n^c) steps, and outputting the result of the simulated TM. We see that log_2(2^(n^c)) steps, which is thus polynomial with respect to the length of our input, meaning that this reduction can be completed in polynomial time as the value of c is known, so we only have to determine the length of the input which can be done in polynomial time.

Thus, because we have a polynomial time reduction from A to B, by our assumption, we can do a polynomial time reduction from B to A. So we have a polynomial time reduction from a P-Complete problem to an EXPTIME-Complete problem.

However, from the book, we know that EXP is a strict superset of P (this follows from the time hierarchy theorem). Thus, we have a contradiction. Thus, the claim is false.

3:
Suppose we have a TM with a non-unary alphabet.

The definition of NTIME is linear time non-deterministic. If each random number generation draws from a sample of size k, we must have that the number of bits to represent asignle sample is at most log(k). So if we draw a number A of size log(k), we must require at least log(k) work. So if we have linear time, non-deterministic work and we draw some series of samples to take this linear amount of work (c*n), we must draw at most c*n/(log(k)) samples, where c is some constant, for the branch that correctly solves the problem.   We thus have k^(cn/log(k)) total work if we were to exhaustively try out every sample in our pool.

Because our TM has a non-unary alphabet, we have k <= c^(log(k)), where c is the number of characters in our alphabet.

We have 2^(log(k)*(cn/log(k))) <= 2^cn.

Thus, NTIME(n) is a subset of E.

4:

```
bool NOTAPERM(int& X) {

for (int i=0; i<n; i++) {
        for (j=0; j<n; j++) {
                if (i == x[j]) {
                        break;
                }
                else if (j == (n-1)) {
                        return ACCEPT
                }
        }
}
RETURN REJECT

}
```

We see that the above checks every possible pair (i,X[j]) for each value possible for j and if it cannot find a match j for any j, we reject. We see that this is correct because if such a j cannot be found for a given i, we must not have a permutation because at least one of the correct numbers would be missing.