Problem 3.1

1. → int n;
```
scanf("%d", &n); //Assuming n > 0
int A[n];
int smallest = A[0];
for(int i=0; i<n; ++i)
{
    if(A[i]<smallest)
        smallest = A[i];
}
```
→I must always check every element so it has to be linear.

2. → extern int x; //x is defined somewhere else
```
int find(int s, int e)
{
        if(s+1==e)
            return s;
        int m=(s+e)/2;
        if(m==x)
            return m;
        else if(m>x)
            return find(m+1, e);
        else return find(s, m);
 }
```
→In each step the search size is reduced by half. It avoids the most number of unnecessary checks. Logarithmic Time. O(lg(n)).

3. → string str="";
```
for(int i=0; i<n; ++i)
  for(int j=32; j<=126; ++j)//32 till 126 are valid password characters
        if(pass[i]==(char)j)
        {
            str+=(char)j;
            break;
        }
```
→I must always check every character and get the correct one. It can't be avoided.
(126-32)n=94n=O(n)=linear

Problem 3.2

The complexity class is linear. Because in each step we decrease the size of the string by one and we finish when it is 0. The rest are all constant times.

Problem 3.3

1→X = 2^{n-1}
0→n = n

I can't avoid not checking 2^{n-1} numbers. Therefore it is O(2^n) which is not polynomial.