```
//problem 1
bool has_dups(node *head) {
    node *temp;
    temp = head;
    node* temp1;
    while(temp->next!=NULL){
        temp1=temp->next;
        while(temp1->next!=NULL) {
            if (temp->val == temp1->val) {
                return true;
            }
            temp1 = temp1->next;
        }
        if (temp->val == temp1->val) {
            return true;
        }
        temp=temp->next;
    }
    return false;
}
```

```
//problem 2
/*
 * 2n-1 will always be odd. Adding odd number will always get you a perfect square
 * eg 1+3 = 4  1+3+5 = 9 and so on.
 * my function simulates this.
 */
int rec(int n){

    if(n==1){
        return n;
    }
    return rec(n-1)+n+n-1;
}
```

```
//problem 3
/*
a.) this code returns an int when im pretty sure it wants to return an array of ints.
 this code is meant to return a copy of an array and does not do that, so it will cause an error if you try to assign an
 array to the value that this function returns.

b.) If the programer was able to use full reasources, with an ide and able to run and test their code, 0/10 points
 but, if they hand wrote to code, definitely 10/10, no points taken off because it is easy to miss a * when hand writting code
 if they where typeing into a textbox for blackboard or google forms, then 8/10, still very easy to miss without an ide
 astriks do too many things in c, and it is an error that is easily caught and fixed if you can run your code. The person
 knows what they are doing and can program so they should get minimal points taken off if in any sort of environment where
 they have to write code without being able to compile and run it.


c.)
*/
int* clone_array(int *a, int n) {
    int *b;
    int i;

    for(i=0; i<n; i++) {
        b[i] = a[i];
    }
    return b;
}
```

n	# ticks
0	0
1	2
2	6
3	12
4	20

for all $n \geq 0$ for n

i) $n + n^2$ ticks printed

The first for loop will print 1 tick for every n

The second nested for loop will print n^2 ticks for every n.

$n = 4$ $\quad 4 + 4^2 = 20$

n	# of ticks
0	1
1	3
2	7
3	15
4	29

1

for all $n \geq 0$ calling foo(n) results in $2^{n+1}-1$ ticks printed

base case $n=0$ foo(0) only 1 tick

if(n>0) false

foo(0) = 1

$a(1) = foo$ $a_0 = 1$

hypothesis $2a_{k-1}+1 = a_k$

$a_k = 2^{n+1}+1$

...

prove $2^{k+2}+1 = a_{(k+1)}$

$a_{k+1} = 2a_k + 1$

$2(2^{n+1}-1)+1$

$2^{n+2}-2+1$

$2^{n+2}-1$

```
//problem 6
/*
   The algorithm never checks to see if a number is used more than once. Check for duplicates in the array and if there
   are duplicates then return false.
 */
// array row[] is assumed to be of length at least 9
bool sudoku_row_ok(int row[]) {
    int sum=0;
    int i;

    for(i=0; i<9; i++) {
        if(row[i] < 1 || row[i] > 9)
            return false;            // out of range
        sum += row[i];
    }
    //check for number duplicates
    int j;

    for(i=0; i<9; i++) {
        for(j=i+1; j<9; j++) {
            if(row[i] == row[j])
                return false;
        }
    }

    if(sum == 45)     // notice:  1+2+3+4+5+6+7+8+9 = 45
        return true;
    else
        return false;
}
```