Recursive solution (write down, if any, assumptions about the problem):

```c
binPrint ( pointer to first node of tree )
{
    print ("ptr -> info");
    if (ptr1 != NULL) binPrint(ptr1);
    if (ptr2 != NULL) binPrint(ptr2);
    else print("End of...");
}

Print out a binary tree.
```
Recursive solution (write down, if any, assumptions about the problem):

```c
INT N;

void rotate(int n) { 
    if (n == 0) print("Done");
    else {
        prev = head;
        current = prev -> next;
        while (current -> next != NULL) {
            prev = prev -> next;
            current = current -> next;
        }
        current -> next = head;
        head = current;
        prev -> next = NULL;
        rotate(n - 1);
    }
}
```

Assume that the list is a linked list.

rotate a list left by n
Recursive solution (write down, if any, assumptions about the problem):

```c
int recursive(int n)
{
    if (n == 0) return 0;
    else
    {
        int sum = n % 10;
        sum += recursive(n / 10);
        return sum;
    }
}
```

Compute the sum of digits of an integer.
Recursive solution (write down, if any, assumptions about the problem):

```
assume punctuation, symbols, spaces have been removed
assume all lowercase

bool palindrome (string s) {
    if (s == "") {
        return true;
    }
    else if (first == last) {
        remove first;
        if (s != "") {
            remove last;
            palindrome (s);
        }
        else
            return false;
    }

Determine whether a string is a palindrome.
```
Recursive solution (write down, if any, assumptions about the problem):

```c
void SweetFunction(int[] input, int[] compose)
{
    if (sizeof(input) < 1)
        print "Error: Input is too small."
    else
        for (int i = 0; i < sizeof(compose); i++)
            if (input[0] == compose[i])
                print compose[i],

    SweetFunction(input without first element, compose);
}
```

Obtain the intersection of 2 lists
Recursive solution (write down, if any, assumptions about the problem):

```c
binary(int n) {
    if (n == 0)
        print "Done";
    else {
        binary(n/2);
        print(n % 2);
    }
}
```

Convert a decimal # to a binary #.

```
binary(12)  
|  
| bin(6) - 0  
| bin(3) - 0  
| bin(1) - 1  
| bin(0)  

1100
```