2.4 Format of a recursive function

```
if (test for a base case)
    return (base case value)
else if (test for another base case)
    return (other base case value)
else // recursive case
    return (some work and recursive call)
```

(i) Calculate factorial of a positive integer

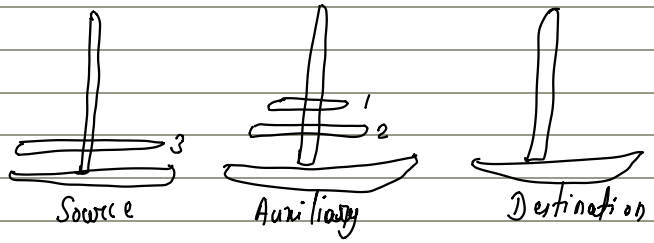
```
int fact(n){
    if (n==0)
        return 1;
    else if (n==1)
        return 1;
    else
        return n * fact(n-1);
}
```

2.9 Recursion: Problems and solutions

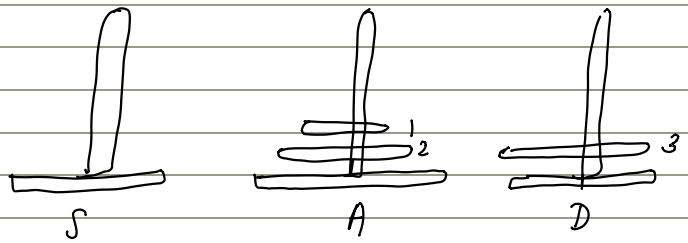
P1. Towers of Hanoi

Algorithms:

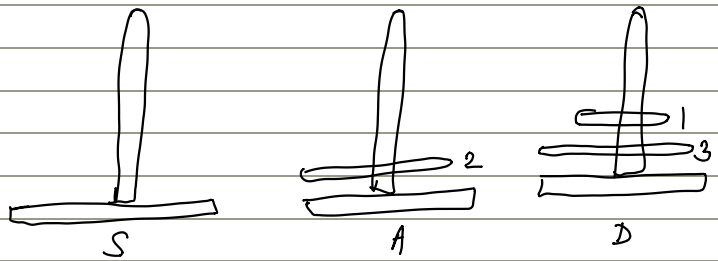
(i) Move the top n-1 disks from source to auxiliary tower.



(ii) Move the n^{th} disk from source to destination tower



(iii) Move the n-1 disks from auxiliary tower to destination tower



Code:

```
void towers_of_hanoi(int n, source, auxiliary, destination){
    if (n==1){
        cout<<"Move disk 1 from "<<source<<"tower to"
            <<destination<<"tower"<<endl;
        return;
    }
    else{
        // Move top n-1 disks from source to auxiliary tower
        towers_of_hanoi(n-1, source, destination, auxiliary);

        // Move nth disk from source to destination tower
        cout<<"Move disk "<<n<<"from "<<source<<"tower to"
            <<destination<<"tower"<<endl;

        // Move n-1 disks from auxiliary to destination tower
        towers_of_hanoi(n-1, auxiliary, source, destination);
    }
}
```

P2. To check whether an array is sorted

Code:

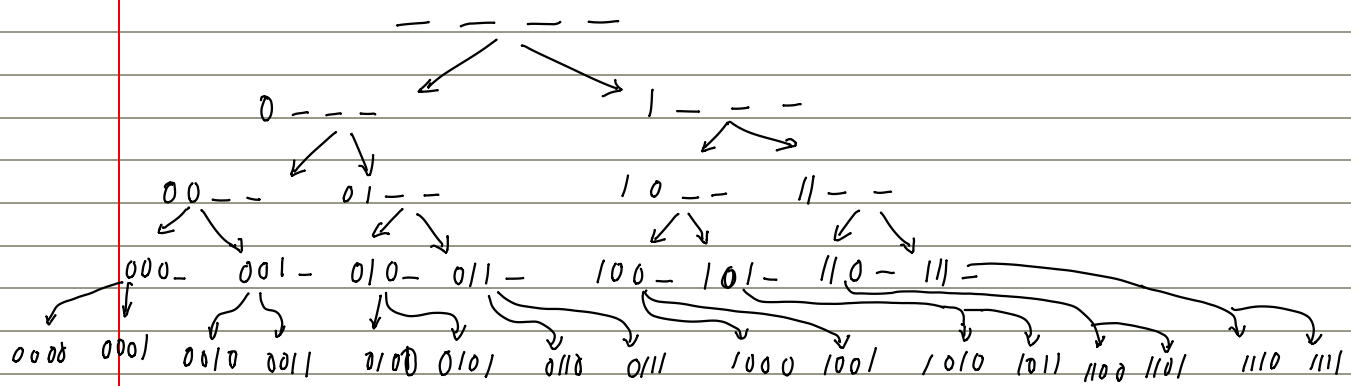
```
bool is_array_sorted(int arr[], int n){
    if (n==1){
        return true;
    else{
        return (arr[n-1]> arr[n-2])? is_array_sorted(arr, n-1): false;
    }
}
```

2.12 Backtracking: Problems and solutions

P3. Generate all strings of n bits

Algorithm:

Ex: if n = 4



Code :

```
void generate-n-bits (int i, bool arr[], int n){
    if (i==n){
        for (int j=0; j<n; j++){
            cout << arr[j] << " ";
        }
        return j
    }

    else{
        arr[i] = 0;
        generate-n-bits (i+1, arr, n);
        arr[i] = 1;
        generate-n-bits (i+1, arr, n);
    }
```