Recursion - III

Permutation

To print all the permutations of a string.

Idea: for each character s[i] in the given string, we add a character in the ans string and then solve solve s.substr(0,i) + s.substr(i+1)

Sample Input:

ABC

Sample Output:

ABC

ACB

BAC

BCA

CAB

CBA

Time Complexity: $O(N*2^n)$

Space Complexity: $O(2^n)$

```
void permutation(string s, string ans) {

    if (s.length() == 0) {
        cout << ans << endl;
        return;
    }

    for (int i = 0; i < s.length(); i++) {
        char ch = s[i];
        string ros = s.substr(0, i) + s.substr(i + 1);

        permutation(ros, ans + ch);
    }
}
```

permutation(s, “”) will give the required answer

CountPaths

Find the number of ways to reach e from s.

Idea:

We have 6 ways to go forward (1,2,3,4,5,6).

At the starting point s,

Current answer = countPath(s+1,e) + countPath(s+2,e) + countPath(s+3,e) + countPath(s+4,e) + countPath(s+5,e) + countPath(s+6,e)

Time Complexity: $O(2^n)$

Space Complexity: $O(2^n)$

```
int countPath(int s, int e) {
    if (s == e) {
        return 1;
    }
    if (s > e) {
        return 0;
    }
    int count = 0;
    for (int i = 1; i <= 6; i++) {
        count += countPath(s + i, e);
    }
    return count;
}
```

CountPathMaze

Given a 2D grid, find the number of ways to reach (n-1, n-1).

You can go to (i,j) from (i-1,j) and (i,j-1).

Time Complexity: $O(2^n)$

Space Complexity: $O(2^n)$

```
int countPathMaze(int n, int i, int j) {
    if (i == n - 1 && j == n - 1) {
        return 1;
    }
    if (i >= n || j >= n) {
        return 0;
    }

    return countPathMaze(n, i + 1, j) +
           countPathMaze(n, i, j + 1);
}
```