Minimum Path Sum:

```
class Solution {
public:
    int minimumPathSum(int i,int j,int m,int
n,vector<vector<int>>&grid,vector<vector<int>>&dp){
        //base case
        if(i==m-1&&j==n-1){
            return grid[i][j];
        }
        //edge case
        if(i>=m||j>=n){
            return 1e9;
        }
        if(dp[i][j]!=-1){
            return dp[i][j];
        }
        //down
        int down=grid[i][j]+minimumPathSum(i+1,j,m,n,grid,dp);

        //right
        int right=grid[i][j]+minimumPathSum(i,j+1,m,n,grid,dp);

        return dp[i][j]=min(down,right);
    }
    int minPathSum(vector<vector<int>>& grid) {
        int m=grid.size();
        int n=grid[0].size();
        int i=0;
        int j=0;
        vector<vector<int>>dp(m,vector<int>(n,-1));
        int ans=minimumPathSum(i,j,m,n,grid,dp);
        return ans;
    }
};
```

Triangle:

```
class Solution {
public:
    int minimumPathSum(int i,int j,int m,int
n,vector<vector<int>>&triangle, vector<vector<int>>&dp){
        //base case
        if(i==m-1){
            return triangle[i][j];
        }
        if(dp[i][j]!=-1){
            return dp[i][j];
        }
        //down
        int d=triangle[i][j]+minimumPathSum(i+1,j,m,n,triangle,dp);


        //diagonal right

        int dr=triangle[i][j]+minimumPathSum(i+1,j+1,m,n,triangle,dp);

        return dp[i][j]=min(d,dr);
    }
    int minimumTotal(vector<vector<int>>& triangle) {
        int m=triangle.size();
        int n=m;
        int i=0;
        int j=0;
        vector<vector<int>>dp(m,vector<int>(n,-1));
        return minimumPathSum(i,j,m,n,triangle,dp);
    }
};
```

Coin Change:

```
class Solution {
public:

    int minimumAmount(int i,vector<int>&coins,int
amount,vector<vector<int>>&dp){
        if(i==coins.size()){
           if(amount==0){
            return 0;
           }
           else{
            return 1e9;
           }
        }
        if(dp[i][amount]!=-1){
            return dp[i][amount];
        }
        //pick
        int pick=1e9;
        if(coins[i]<=amount){
            pick=1+minimumAmount(i,coins,amount-coins[i],dp);
        }
        //not pick
        int notpick=0+minimumAmount(i+1,coins,amount,dp);

        return dp[i][amount]=min(pick,notpick);
    }
    int coinChange(vector<int>& coins, int amount) {
        int i=0;
        vector<vector<int>>dp(coins.size(),vector<int>(amount+1,-1));
        int ans=minimumAmount(i,coins,amount,dp);
        if(ans==1e9){
            return -1;
        }
        return ans;
    }
};
```