Course Schedule

```
#include <vector>
#include <queue>
using namespace std;

class Solution {
public:
    bool canFinish(int num, vector<vector<int>>& prereqs) {
        vector<vector<int>> graph(num);
        vector<int> indeg(num, 0);

        for (const auto& p : prereqs) {
            int course = p[0];
            int prereq = p[1];
            graph[prereq].push_back(course);
            indeg[course]++;
        }

        queue<int> q;
        for (int i = 0; i < num; i++) {
            if (indeg[i] == 0) q.push(i);
        }

        int count = 0;
        while (!q.empty()) {
            int curr = q.front();
            q.pop();
            count++;
            for (int neighbor : graph[curr]) {
                indeg[neighbor]--;
                if (indeg[neighbor] == 0) q.push(neighbor);
            }
        }

        return count == num;
    }
};
```

Ancient realm of numeria

```
#include <iostream>
#include <vector>
using namespace std;

const int MOD = 1e9 + 7;

int countSequences(int S) {
    // DP array to store the number of valid sequences for each sum from 0 to S
    vector<int> dp(S + 1, 0);

    // Base case: there's 1 way to form sum 0 (using no numbers)
    dp[0] = 1;

    // Fill the DP table for each sum from 3 to S
    for (int i = 3; i <= S; ++i) {
        for (int j = 3; j <= i; ++j) {
            dp[i] = (dp[i] + dp[i - j]) % MOD;
        }
    }

    // Return the result for sum S
    return dp[S];
}

int main() {
    int S;
    cout << "Enter the target sum S: ";
    cin >> S;

    // Calculate and output the number of valid sequences
    int result = countSequences(S);
    cout << result << endl;

    return 0;
}
```