과제 #6 타일 문양 - 201921438 조현태

1) 소스코드

```
#include <stdio.h>
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>
using namespace std;
// n의 최대범위
#define MAX 1000001

// 함수 선언
void print_tile();

// 0 < n <= 1000000
int n;

// 2차원 배열
int dp[MAX][3] = {0, };

// 3차원 배열 -> 2차원 배열에는 개수만 들어가기 때문에 3차원배열에 문자열을 입력받음.
string dp_tile[31][3][585] = {"0", };

int main()
{
	cin >> n;

	// 초기값 설정
	dp[1][1] = 1;
	dp[2][1] = 1;
	dp[2][2] = 1;

	if (n <= 30)
	{
		// 문자열을 위한 초기값 설정
		dp_tile[1][1][0] = "|";
		dp_tile[2][1][0] = "||";
		dp_tile[2][2][0] = "==";
		dp_tile[3][1][0] = "==|";
		dp_tile[3][2][0] = "|==";
	}
```

```
	// 점화식을 이용한 DP풀이
	for (int i = 3; i <= n; i++)
	{
		// "|"가 오려면 전에 "="가 오던가 "= |"가 와야함.
		dp[i][1] = (dp[i - 1][2] + dp[i - 2][2]) % 20201212;
		// "="가 오려면 2칸전에 "|"가 와야함.
		dp[i][2] = dp[i - 2][1] % 20201212;

		// n이 30이하 일 경우, 타일을 출력해야하므로 string벡터에 따로 처리
		if ((n <= 30) && (i > 3))
		{
			int k = 0;
			for (int j = 0; j < dp[i - 1][2]; j++)
			{
				dp_tile[i][1][k] = dp_tile[i - 1][2][j] + "|";
				k++;
			}
			for (int j = 0; j < dp[i - 2][2]; j++)
			{
				dp_tile[i][1][k] = dp_tile[i - 2][2][j] + "||";
				k++;
			}
			for (int j = 0; j < dp[i - 2][1]; j++)
				dp_tile[i][2][j] = dp_tile[i - 2][1][j] + "==";
		}
	}

	// 정답 출력
	cout << (dp[n][1] + dp[n][2]) % 20201212 << endl;

	// n이 30이하 일 경우, 타일 출력
	if (n <= 30)
		print_tile();

	return 0;
}

// 타일 출력 함수
void print_tile()
{
	// 두 벡터 합치기
	vector <string> DP1 (begin(dp_tile[n][1]), end(dp_tile[n][1]));
	vector <string> DP2 (begin(dp_tile[n][2]), end(dp_tile[n][2]));
```

```
        DP1.insert(DP1.end(), DP2.begin(), DP2.end());

        // 문자열을 기준을 오름차순으로
        sort(DP1.begin(), DP1.end());

        // 공백 제거
        DP1.erase(unique(DP1.begin(), DP1.end()), DP1.end());

        // 타일 문양 출력
        for (int i = 1; i < DP1.size(); i++)
                cout << DP1[i] << endl;

        return;
}
```

2) 문제 설명

이 문제는 n이 주어질 때, 타일을 나열하는 문제입니다.
일단 n의 범위가 1000000이고 조건에서 “|”이 연속으로 3번 오지 못하고
“==”이 연속으로 오지 못한다는 것으로 보아 DP를 사용한다고 짐작했습니다.
따라서 n을 키워나가면 점화식을 찾으려고 했습니다.

n = 1 : 1
|
n = 2 : 2
||, ==
n = 3 : 2
==|, |==
n = 4 : 3
==||, |==|, ||==
n = 5 : 4
==|==, |==||, ||==|

총 길이는 2 x N이고 “|”은 2 x 1, “==”은 2 x 2이므로 1과 2로 나타냈습니다.

	1 (|)	2 (==)
1	1 (|)	0
2	1 (||)	1 (==)
3	1 (==|)	1 (|==)
4	2 (==||), (|==|)	1 (||==)
5	2 (|==||), (||==|)	1 (==|==)
6	2	2
7	3	2

마지막에 “|”이 올 경우는 “==” + “|” 아니면 “==” + “||” 이고
마지막에 “==”가 올 경우는 “|” + “==” 임을 알 수 있습니다.
따라서 이를 점화식으로 표현하면,
dp[i][1] = dp[i - 1][2] + dp[i - 2][2]
dp[i][2] = dp[i - 2][1] 임을 알 수 있습니다.

즉, n일 때 타일 문양의 개수는 dp[i][1] + dp[i][2]입니다.

이어서 n <= 30일 때, 모든 타일 문양의 모양을 출력하라고 했으므로
각각의 케이스마다 문자열로 받을 수 있는 배열을 만들었습니다.

이때, 제가 만든 점화식은 2차 배열에 개수를 넣은 것이므로
개수를 모두 문자열로 출력하기 위해서는 3차 배열이 필요합니다.
이때 테스트케이스에서 n=31일 때 585라 하였으므로
3차 배열 dp_tile[31][3][585]로 설정했습니다.

	1	2
1	|	0
2	||	==
3	==|	|==
4	|==|, ==||	||==
5	||==|, |==||	==|==
6	||==||, ==|==|	|==|==, ==||==
7	|==|==|, ==||==|, ==|==||	||==|==, |==||==

```
int k = 0;
for (int j = 0; j < dp[i - 1][2]; j++)
{
        dp_tile[i][1][k] = dp_tile[i - 1][2][j] + "|";
        k++;
}
for (int j = 0; j < dp[i - 2][2]; j++)
{
        dp_tile[i][1][k] = dp_tile[i - 2][2][j] + "||";
        k++;
}
for (int j = 0; j < dp[i - 2][1]; j++)
        dp_tile[i][2][j] = dp_tile[i - 2][1][j] + "==";
```

문제에서 모든 문자열을 사전식 순서로 출력하라고 하였으므로 (“=”가 “|”보다 앞섬.)
dp[n][1]과 dp[n][2]를 합치고 오름차순으로 정리하여 출력했습니다.