https://www.acmicpc.net/problem/8111

구사과님이 문제에 주어진 조건에 만족하는 수를 좋아하신다고 합니다.
자연수 N이 주어질 때 N의 배수 중에서 구사과님이 좋아하시는 수를 아무거나 출력하는 문제입니다.

풀이

완전 탐색으로 100자릿수에 0,1을 넣어보면 2^100 가지의 경우가 있으니 안됩니다.
N의 배수를 확인하며 0과 1로 이루어진 수를 찾을 수도 없습니다.
그래서 0과 1로 이루어진 수를 만들면서 N으로 나누어 N의 배수인지 확인하겠습니다.

완전 탐색과 뭐가 다른가요?

수가 N의 배수인지 확인을 하기 위해서는 나머지 연산을 했을 때 0이면 됩니다.
그래서 그 수를 나눌 필요 없이 나머지들로 계산을 하면 됩니다.
예를 들면, (1111 * 10 + 1) % 7 = 2, ((1111 % 7) * 10 + 1) % 7 = 2 이런 식으로 모듈러 연산은 분배 법칙이 적용합니다.

그래서

1.(A * 10 + 1) % N

2. (A * 10) % N

이 두 개를 연산하며 나머지가 0일 때까지 연산하면 됩니다.

1부터 시작하니 1을 기준으로 BFS를 돌리면 가능한 수 중에 가장 짧은 수를 구할 수 있습니다.

그러면 그 수가 길이가 100이 넘어가면 BRAK을 출력하면 되겠죠?

BFS를 하면서 현재 무슨 수를 추가했는지 저장을 하게 되면 그 수를 쉽게 알아낼 수 있습니다.

그런데 이 문제가 사실 그러한 수는 꼭 있다고 합니다.

0이 아닌 사이클이 돌게 되면 아예 불가능한데

https://kimcodingvv.github.io/BOJ-1612/

[BOJ | 백준] 1612번: 가지고 노는 1

BOJ 1612 가지고 노는 1

kimcodingvv.github.io

이 글에서 1로만 이루어진 수는 2,5배수를 제외하고 가능하다고 했는데 0으로도 수를 만들 수 있으니 모두 가능하다는 것을 알 수 있습니다.

2만의 길이를 가진 수는 가능하다는 것을 알았는데 수의 길이가 100이하인 경우가 가능한지는 잘 모르겠습니다...

알게 될 기회가 있다면 글을 수정하여 올리겠습니다.

소스 코드

```cpp
#include <bits/stdc++.h>
#define FIO ios::sync_with_stdio(0), cin.tie(0),cout.tie(0)
#define FOR(i,a,b) for(int i=a;i<=b;i++)
#define pii pair <int,int>
#define fs first
#define sd second
using namespace std;

int main(){
    FIO;
    int t; cin >> t ;while(t--){
        int n; cin >> n;
        if(n == 1) {
            cout << 1 << '\n';
            continue;
        }
        queue <int> q;
        vector <pii> chk(20005, {-1,-1});
        q.emplace(1); chk[1] = {1,-1};
        while(q.size()){
            int now = q.front(); q.pop();
            int a = (now * 10 + 1) % n, b = (now * 10) % n;
            if(chk[a].fs == -1) chk[a] = {1, now}, q.emplace(a);
            if(chk[b].fs == -1) chk[b] = {0, now}, q.emplace(b);
            if(!a || !b) break;
        }
        stack <int> s;
        for(int i = 0; i != -1; i = chk[i].sd)
            s.emplace(chk[i].fs);
        if((int)s.size() > 100) cout << "BRAK";
        else while(s.size()) {
            cout << s.top();
            s.pop();
        }
        cout << '\n';
    }
    return 0;
}
```