Write a program which, given an integer nn as input, will produce a mathematical expression whose solution is nn. The solution is restricted to using exactly four 44’s and exactly three of the binary operations selected from the set {∗,+,−,/}{∗,+,−,/}. The number 44 is the ONLY number you can use. You are not allowed to concatenate fours to generate other numbers, such as 4444 or 444444.

For example given n=0n=0, a solution is 4∗4−4∗4=04∗4−4∗4=0. Given n=7n=7, a solution is 4+4−4 / 4=74+4−4 / 4=7. Division is considered truncating integer division, so that 1/41/4 is 00 (instead of 0.250.25). Assume the usual precedence of operations so that 4+4∗4=204+4∗4=20, not 3232. Not all integer inputs have solutions using four 44’s with the aforementioned restrictions (consider n=11n=11).

*Hint: Using your forehead and some forethought should make an answer forthcoming. When in doubt use the fourth.*

**Input**

Input begins with an integer 1≤m≤10001≤m≤1000, indicating the number of test cases that follow. Each of the next mm lines contain exactly one integer value for nn in the range −1000000≤n≤1000000−1000000≤n≤1000000.

**Output**

For each test case print one line of output containing either an equation using four 44’s to reach the target number or the phrase no solution. Print the equation following the format of the sample output; use spaces to separate the numbers and symbols printed. If there is more than one such equation which evaluates to the target integer, print any one of them.

|  |  |
| --- | --- |
| **Sample Input 1** | **Sample Output 1** |
| 5  9  0  7  11  24 | 4 + 4 + 4 / 4 = 9  4 \* 4 - 4 \* 4 = 0  4 + 4 - 4 / 4 = 7  no solution  4 \* 4 + 4 + 4 = 24 |

很多人用手动打表过了……嗯就我一个人用了中缀转后缀……呵呵呵呵呵。

#include <iostream>

#include<sstream>

#include<iomanip>

#include<string>

#include<vector>

#include<stack>

#include<queue>

#include<algorithm>

#include<map>

#define hash 997

#define MAX 100005

#define ll long long

using namespace std;

struct answer {

int num;

vector<char>list;

};

vector<answer>answer\_list;

vector<char>method = { '+','-','\*','/' };

int process(vector<char>temp)

{

string origin = "4";

string result;

int total = 0;

for (int i = 0; i < temp.size(); i++)//?

{

origin.push\_back(temp[i]);

origin.push\_back('4');

}

stack<char>S1;

stack<int>S2;

for (int i = 0; i < origin.size(); i++)

{

if (origin[i] >= '0'&&origin[i] <= '9')

result.push\_back(origin[i]);

else

{

if (S1.empty())

S1.push(origin[i]);

else

{

if (origin[i] == '+' || origin[i] == '-')

{

while (!S1.empty())

{

result.push\_back(S1.top());

S1.pop();

}

S1.push(origin[i]);

}

else

{

while (!S1.empty())

{

if (S1.top() == '+' || S1.top() == '-')

break;

result.push\_back(S1.top());

S1.pop();

}

S1.push(origin[i]);

}

}

}

}

while (!S1.empty())

{

result.push\_back(S1.top());

S1.pop();

}

for (int i = 0; i < result.size(); i++)

{

if (result[i] >= '0'&&result[i] <= '9')

{

S2.push(result[i]-'0');

}

else

{

int total,temp2 = S2.top();

S2.pop();

int temp1 = S2.top();

S2.pop();

if (result[i] == '+')

total = temp1 + temp2;

if (result[i] == '-')

total = temp1 - temp2;

if (result[i] == '\*')

total = temp1 \* temp2;

if (result[i] == '/')

total = temp1 / temp2;

S2.push(total);

}

}

return S2.top();

}

int main()

{

vector<char>temp;

for (int i = 0; i < 4; i++)

{

temp.push\_back(method[i]);

for (int j = 0; j < 4; j++)

{

temp.push\_back(method[j]);

for (int k = 0; k < 4; k++)

{

temp.push\_back(method[k]);

answer answer\_temp;

answer\_temp.num=process(temp);

answer\_temp.list = temp;

answer\_list.push\_back(answer\_temp);

temp.pop\_back();

}

temp.pop\_back();

}

temp.pop\_back();

}

int time;

cin >> time;

while (time--)

{

int temp;

cin >> temp;

int flag = 1;

for (int i = 0; i < answer\_list.size(); i++)

{

if (temp == answer\_list[i].num)

{

flag = 0;

printf("4 %c 4 %c 4 %c 4 = %d\n", answer\_list[i].list[0], answer\_list[i].list[1], answer\_list[i].list[2], temp);

break;

}

}

if (flag)

printf("no solution\n");

}

}