Problem A

Jeszcze więcej NikomuNiePotrzebnychFunkcji

X OTWARTE MISTRZOSTWA OPOLA W PROGRAMOWANIU ZESPOŁOWYM

Zawodnicy VII Otwartych Mistrzostw Opola w Programowaniu Zespołowym zmierzyli się (i to z całkiem dobrym rezultatem!) z dość oryginalnie nazwaną funkcję rekurencyjną, której pierwotnego przeznaczenia nikt nie umiał się domyślić. Cóż… odnaleziono więcej tych dziwacznych tworów, co więcej tym razem, odwołują się do siebie nawzajem. Czy uda wam się zgłębić ich tajemnice?

```
int nikomuNiePotrzebnaFunkcja1 (int n)
{
    if (n % 6 == 0)
    {
        int x = n / 3;
        n = n * 2;
        n = n / 3;
        n = n + x;

        return nikomuNiePotrzebnaFunkcja2 (n);
    }
    else
    {
        n = n + 2;

        return nikomuNiePotrzebnaFunkcja3 (n);
    }

    return 0;
}

int nikomuNiePotrzebnaFunkcja2 (int n)
{
    int x = n * 2;
    n = n * 3 - x;
    n = n + (n - (n - 1));
    n = n - 7;

    if (n < 10)
    {
        return n;
    }
    else
    {
        return nikomuNiePotrzebnaFunkcja1 (n);
    }

    return 0;
}

int nikomuNiePotrzebnaFunkcja3 (int n)
{
    n = n * 3 - (n + (n - 1) + 1);

    while (n % 6 != 0)
    {
        n = n - 1;
    }

    return nikomuNiePotrzebnaFunkcja1 (n);
}
```

Zadanie

Napisz program, który dla zadanego *N* zwróci wynik działania funkcji `nikomuNiePotrzebnaFunkcja1`.

Wejście

Program będzie sprawdzany przy pomocy zestawu *T* testów ($1 \leq T \leq 2000$). Pierwszy wiersz standardowego wejścia zawiera liczbę testów. Od drugiego wiersza umieszczono dane wejściowe dla kolejnych testów. Dane wejściowe dla jednego testu stanowi pojedynczy wiersz zawierający liczbę całkowitą *N* ($-500000 \leq N \leq 500000$).

Wyjście

W kolejnych wierszach standardowego wyjścia należy podać odpowiedzi obliczone dla kolejnych testów. Wynikiem dla jednego testu jest pojedyncza liczba całkowita, której wartość byłaby zwrócona przez funkcję `nikomuNiePotrzebnaFunkcja1` dla zadanego *N*.

Przykład

Wejście

```
5
-8
0
1
4
178342
```

Prawidłowe wyjście

```
-12
-6
-6
0
6
```

Definicje funkcji w języku Pascal

```
function nikomuNiePotrzebnaFunkcja1 (n :  integer) : integer;
var
      x : integer;
 begin
    if n mod 6 = 0 then
    begin
        x := n div 3;
        n := n * 2;
        n := n div 3;
        n := n + x;
        nikomuNiePotrzebnaFunkcja1 := nikomuNiePotrzebnaFunkcja2 (n);
    end else
    begin
        n := n + 2;
        nikomuNiePotrzebnaFunkcja1 := nikomuNiePotrzebnaFunkcja3 (n);
    end;
end;
```

```
function nikomuNiePotrzebnaFunkcja2 (n :  integer) : integer;
var
      x : integer;
 begin
    x := n * 2;
    n := n * 3 - x;
    n := n + (n - (n - 1));
    n := n - 7;

    if n < 10 then
              nikomuNiePotrzebnaFunkcja2 := n
       else
              nikomuNiePotrzebnaFunkcja2 :=  nikomuNiePotrzebnaFunkcja1 (n);
end;


function nikomuNiePotrzebnaFunkcja3 (n :  integer) : integer;
begin
    n := n * 3 - (n + (n - 1) + 1);

    while n mod 6 <> 0 do
    begin
        n := n - 1;
    end;

    nikomuNiePotrzebnaFunkcja3 := nikomuNiePotrzebnaFunkcja1 (n);
end;
```