<BOJ 14888_연산자 끼워넣기>

문제
N개의 수와 N−1개의 연산자가 주어진다. 주어진 수의 순서를 그대로 유지하면서, 주어진 연산자들을 모든 가능한 순서로 조합하여 최댓값과 최솟값을 출력하는 프로그램을 작성

입력
첫째 줄: 수의 개수 N (2 ≤ N ≤ 11)
둘째 줄: N개의 수 (1 ≤ Ai ≤ 100)
셋째 줄: +, -, ×, ÷ 연산자의 개수 (총합 = N - 1)

출력
첫째 줄: 최댓값
둘째 줄: 최솟값

예제 입력 1
2
5 6
0 0 1 0

예제 출력 1
30
30

예제 입력 2
3
3 4 5
1 0 1 0

예제 출력 2
35
17

접근 방법
1. 백트래킹으로 가능한 모든 연산자 조합을 탐색한다.
숫자의 순서를 고정하고 연산자 순서를 바꿔가며 가능한 모든 경우의 수를 계산한다.
2. 재귀 함수 func(k, res)를 통해 현재까지 계산된 결과 res, 다음 숫자 인덱스 k를 변수로 넘긴다.(k=1부터 시작) 남은 연산자 중 하나를 선택해 사용하고 다음 숫자와 계산한 값을 다시 func에 전달한다. 사용한 연산자는 op[]배열에서 하나 줄이고 재귀 호출이 끝나면 op[]++를 통해 다시 복구한다. k==N이면 모든 숫자를 다 사용한 것이므로 결과를 res_max,

res_min과 비교해 갱신하면서 모든 경우를 다 탐색한 뒤에 최댓값과 최솟값을 구해서 출력한다.

```
/*BOJ/14888/연산자 끼워넣기
0ms
70m
*/



#include<bits/stdc++.h>

using namespace std;

int N;
int re;
int op[4];
int num[105];
int re_max = -1000000000;
int re_min = 1000000000;

void func(int k, int res){
	if(k == N){
		re_max = max(re_max, res);
		re_min = min(re_min, res);
		return;

	}
	for(int i = 0; i<4; i++){
		if(op[i]>0){
		op[i]--;
		switch (i){
			case 0:
				func(k+1,res+num[k]);
				break;
			case 1:
				func(k+1,res-num[k]);
				break;
			case 2:
				func(k+1,res*num[k]);
				break;
			default:
				func(k+1,res/num[k]);
		}
```

```
                op[i]++;
                }
        }
}
int main(){
        ios::sync_with_stdio(0);
        cin.tie(0);

        cin>> N;
        for(int i=0;i<N;i++){
                cin>>num[i];
        }
        for(int i=0;i<4;i++){
                cin>>op[i];
        }

        func(1,num[0]);

        cout<<re_max<<"\n";
        cout<<re_min;


        return 0;
}
```