Validator 제작

> 출제자: 노세윤
> 정해: 구현

풀이

여러가지 방법으로 풀 수 있습니다.

다만 모든 모서리의 길이가 같은지만 확인한다면 틀리게 됩니다. 밑면이 마름모인 사각기둥도 모든 모서리 길이가 같을 수 있기 때문입니다.

제가 구현한 알고리즘은 다음과 같습니다.

- 중복되는 점이 있다면 "no"를 출력합니다.
- 아무점이나 한 점을 잡은 뒤 그 점에서 가장 가까운 세 점까지의 거리가 모두 같은지 확인합니다. 만약 다르다면 "no"를 출력합니다.
- 제일 먼저 잡았던 점을 시점으로 하고 가장 가까운 세 점을 각각 종점으로 하는 세 벡터를 구합니다. 세 벡터끼리 내적을 하여 서로 수직인지 확인합니다. 수직이 아니라면 "no"를 출력합니다.
- 제일 먼저 잡았던 점의 위치벡터를 P라고 하고 위에서 구한 세 벡터를 각각 a, b, c라고 하면 0또는 1인 자연수 x, y, z에 대해 $P + x \cdot a + y \cdot b + z \cdot c$를 위치벡터로 하는 점이 반드시 존재해야 합니다. 이것은 std::map 등을 이용하면 쉽게 확인할 수 있습니다. 만약 그 위치에 점이 존재하지 않다면 "no"를 출력합니다.

첨언하자면 벡터끼리의 덧셈, 뺄셈, 내적 등의 연산은 c++의 연산자 오버로딩을 이용하면 쉽게 구현할 수 있습니다.

```
#include <bits/stdc++.h>
using namespace std;

struct A{
    long long x,y,z;
    A operator +(const A&r)const{
        return {x+r.x,y+r.y,z+r.z};
    }
    long long operator *(const A&r)const{
        return (r.x-x)*(r.x-x)+(r.y-y)*(r.y-y)+(r.z-z)*(r.z-z);
    }
    bool operator <(const A&r)const{
        if(x!=r.x)return x<r.x;
        if(y!=r.y)return y<r.y;
        return z<r.z;
    }
    A operator -(const A&r)const{
        return {x-r.x,y-r.y,z-r.z};
    }
```

```
    long long operator /(const A&r)const{
        return x*r.x+y*r.y+z*r.z;
    }
};
void f(){
    cout<<"no";
    exit(0);
}

int main() {

    vector<A> a(8);
    map<A,int> mp;
    for(int i=0 ; i<8; i++){
        cin>>a[i].x>>a[i].y>>a[i].z;
        if(mp[a[i]])f();
        mp[a[i]]=1;
    }

    A q,w,e;
    long long d=1e18;
    for(int i=1 ; i<8 ; i++){
        d=min(d,a[0]*a[i]);
    }
    if(d==0)f();
    int cnt=0;

    for(int i=1 ; i<8 ; i++){
        if(a[i]*a[0]==d){
            if(cnt==0){
                q=a[i]-a[0];
            }
            else if(cnt==1){
                w=a[i]-a[0];
            }
            else if(cnt==2){
                e=a[i]-a[0];
            }
            else{
                f();
            }
            cnt++;
        }
    }

    if(cnt<3)f();
    if(q/w==0 && w/e==0 && e/q==0);
    else f();
    A r=a[0];
```

```
    if(mp[r] && mp[r+q] && mp[r+w] && mp[r+e] && mp[r+q+w] && mp[r+q+e] &&
mp[r+w+e] && mp[r+q+w+e]);
    else f();

    cout<<"yes";
    return 0;
}
```