```cpp
#include<bits/stdc++.h>
using namespace std;
using Int = long long;
//BEGIN CUT HERE

#define EPS (1e-10)
#define equals(a,b) (fabs((a)-(b)) < EPS)
#define PI 3.141592653589793238
struct Point3D{
  double x,y,z;
  Point3D(){}
  Point3D(double x,double y,double z):x(x),y(y),z(z){}
  Point3D operator+(Point3D p) {return Point3D(x+p.x,y+p.y,z+p.z);}
  Point3D operator-(Point3D p) {return Point3D(x-p.x,y-p.y,z-p.z);}
  Point3D operator*(double k){return Point3D(x*k,y*k,z*k);}
  Point3D operator/(double k){return Point3D(x/k,y/k,z/k);}
  Point3D operator*(Point3D p){
    return Point3D(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);
  }
  double operator^(Point3D p){
    return x*p.x+y*p.y+z*p.z;
  }
  double norm(){return x*x+y*y+z*z;}
  double abs(){return sqrt(norm());}
  bool operator < (const Point3D &p) const{
    if(x!=p.x) return x<p.x;
    if(y!=p.y) return y<p.y;
    return z<p.z;
  }
  bool operator == (const Point3D &p) const{
    return fabs(x-p.x)<EPS && fabs(y-p.y)<EPS && fabs(z-p.z)<EPS;
  }
};
istream &operator >> (istream &is,Point3D &p){
  is>>p.x>>p.y>>p.z;
  return is;
}
ostream &operator << (ostream &os,Point3D p){
  os<<fixed<<setprecision(12)<<p.x<<" "<<p.y<<" "<<p.z;
  return os;
}

typedef Point3D Vector3D;
typedef vector<Point3D> Polygon3D;

struct Segment3D{
  Point3D p1,p2;
  Segment3D(){}
  Segment3D(Point3D p1, Point3D p2):p1(p1),p2(p2){}
};
typedef Segment3D Line3D;

istream &operator >> (istream &is,Segment3D &s){
  is>>s.p1>>s.p2;
  return is;
}

struct Sphere{
  Point3D c;
  double r;
```

```
  Sphere(){}
  Sphere(Point3D c,double r):c(c),r(r){}
};

istream &operator >> (istream &is,Sphere &c){
  is>>c.c>>c.r;
  return is;
}

double norm(Vector3D a){
  return a.x*a.x+a.y*a.y+a.z*a.z;
}
double abs(Vector3D a){
  return sqrt(norm(a));
}
double dot(Vector3D a,Vector3D b){
  return a.x*b.x+a.y*b.y+a.z*b.z;
}
Vector3D cross(Vector3D a,Vector3D b){
  return Vector3D(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x);
}

Point3D project(Line3D l,Point3D p){
  Point3D b=l.p2-l.p1;
  double t=dot(p-l.p1,b)/norm(b);
  return l.p1+b*t;
}

Point3D reflect(Line3D l,Point3D p){
  return p+(project(l,p)-p)*2.0;
}

double getDistanceLP(Line3D l,Point3D p){
  return abs(cross(l.p2-l.p1,p-l.p1)/abs(l.p2-l.p1));
}

double getDistanceSP(Segment3D s,Point3D p){
  if(dot(s.p2-s.p1,p-s.p1) < 0.0 ) return abs(p-s.p1);
  if(dot(s.p1-s.p2,p-s.p2) < 0.0 ) return abs(p-s.p2);
  return getDistanceLP(s,p);
}

bool intersectSC(Segment3D s,Sphere c){
  double d=getDistanceSP(s,c.c);
  if(d>c.r) return 0;
  return !((abs(s.p1-c.c)<=c.r)&&(abs(s.p2-c.c)<=c.r));
}

struct ConvexHull3D{
  struct face{
    Int a,b,c;
    bool ok;
    face(){}
    face(Int a,Int b,Int c,bool ok):a(a),b(b),c(c),ok(ok){}
  };
  Int n,num;
  vector<Point3D> p;
  vector<face> f;
  vector<vector<Int> >  g;

```

```
121   ConvexHull3D(Int n):n(n),p(n),f(n*8),g(n,vector<Int>(n)){}
122
123   void input(){
124     for(Int i=0;i<n;i++) cin>>p[i];
125   }
126
127   double dblcmp(Point3D q,face f){
128     Point3D m=p[f.b]-p[f.a];
129     Point3D n=p[f.c]-p[f.a];
130     Point3D t=q-p[f.a];
131     return (m*n)^t;
132   }
133
134   void deal(Int q,Int a,Int b){
135     Int idx=g[a][b];
136     face add;
137     if(f[idx].ok){
138       if(dblcmp(p[q],f[idx])>EPS) dfs(q,idx);
139       else{
140         add=face(b,a,q,1);
141         g[q][b]=g[a][q]=g[b][a]=num;
142         f[num++]=add;
143       }
144     }
145   }
146
147   void dfs(Int q,Int now){
148     f[now].ok=0;
149     deal(q,f[now].b,f[now].a);
150     deal(q,f[now].c,f[now].b);
151     deal(q,f[now].a,f[now].c);
152   }
153
154   void build(){
155     num=0;
156     if(n<4) return;
157     bool flg=1;
158     for(Int i=1;i<n;i++){
159       if(abs(p[0]-p[i])>EPS){
160         swap(p[1],p[i]);
161         flg=0;
162         break;
163       }
164     }
165     if(flg) return;
166     flg=1;
167     for(Int i=2;i<n;i++){
168       if(abs((p[0]-p[1])*(p[1]-p[i]))>EPS){
169         swap(p[2],p[i]);
170         flg=0;
171         break;
172       }
173     }
174     if(flg) return;
175     flg=1;
176     for(Int i=3;i<n;i++){
177       if(abs(((p[0]-p[1])*(p[1]-p[2]))^(p[0]-p[i]))>EPS){
178         swap(p[3],p[i]);
179         flg=0;
180         break;
```

```
      }
    }
    if(flg) return;
    face add;
    for(Int i=0;i<4;i++){
      add=face((i+1)%4,(i+2)%4,(i+3)%4,1);
      if(dblcmp(p[i],add)>0) swap(add.b,add.c);
      g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=num;
      f[num++]=add;
    }
    for(Int i=4;i<n;i++){
      for(Int j=0;j<num;j++){
        if(f[j].ok&&dblcmp(p[i],f[j])>EPS){
          dfs(i,j);
          break;
        }
      }
    }
    Int tmp=num;
    num=0;
    for(Int i=0;i<tmp;i++)
      if(f[i].ok) f[num++]=f[i];
  }

  double volume(Point3D a,Point3D b,Point3D c,Point3D d){
    return ((b-a)*(c-a))^(d-a);
  }

  bool same(Int s,Int t){
    Point3D &a=p[f[s].a];
    Point3D &b=p[f[s].b];
    Point3D &c=p[f[s].c];
    return  (abs(volume(a,b,c,p[f[t].a]))<EPS)
      &&    (abs(volume(a,b,c,p[f[t].b]))<EPS)
      &&    (abs(volume(a,b,c,p[f[t].c]))<EPS);
  }

  Int polygon(){
    Int res=0;
    for(Int i=0;i<num;i++){
      Int flg=1;
      for(Int j=0;j<i;j++)
        flg&=!same(i,j);
      res+=flg;
    }
    return res;
  }

  Int triangle(){
    return num;
  }

  double area(Point3D a,Point3D b,Point3D c){
    return abs((b-a)*(c-a));
  }

  Point3D cross(Point3D a,Point3D b,Point3D c){
    return Point3D((b.y-a.y)*(c.z-a.z)-(b.z-a.z)*(c.y-a.y),
                   (b.z-a.z)*(c.x-a.x)-(b.x-a.x)*(c.z-a.z),
                   (b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.x));
```

```
  }

  double area(){
    double res=0;
    if(n==3){
      Point3D q=cross(p[0],p[1],p[2]);
      res=abs(q)/2.0;
      return res;
    }
    return res;
    for(Int i=0;i<num;i++)
      res+=area(p[f[i].a],p[f[i].b],p[f[i].c]);
    return res/2.0;
  }
};

//END CUT HERE

signed main(){
  Int n;
  while(cin>>n){
    ConvexHull3D ch(n);
    ch.input();
    ch.build();
    cout<<ch.polygon()<<endl;
  }
  return 0;
}

/*
  verified on 2017/12/31
  http://rhodon.u-aizu.ac.jp:8080/arena/room.jsp?id=3794
*/
```