7-1

太难，不会

7-2

优雅且简单的做法：

欧拉函数

```
int euler_phi(int n) {
  int m = int(sqrt(n + 0.5));
  int ans = n;
  for (int i = 2; i <= m; i++)
    if (n % i == 0) {
      ans = ans / i * (i - 1);
      while (n % i == 0) n /= i;
    }
  if (n > 1) ans = ans / n * (n - 1);
  return ans;
}
```

蒟蒻出题人忘了欧拉函数乱搞的做法：

分解质因数+容斥原理

直接算和n互质的数的数量不太好算，可以先算出不和n互质的数的数量m，则答案就是n-m。

把n分解成质数相乘的形式，即$n=p_1^{x_1}p_2^{x_2}\dots p_k^{x_k}$，对于每个$p_i$,它本身以及它倍数必定和n不互质，所以$\frac{n}{p_i}$就是小于n，且是$p_i$的倍数的数的数量。

但是我们发现直接把所有的$\frac{n}{p_i}$加起来会有重复的部分，比如$15=3\times5$,15既是3的倍数又是5的倍数，则计数的时候15会被重复计数，可以利用容斥原理解决。

注意到$2\times3\times5\times7\times11\times13\times17\times19\times23=223092870>2e8$,则n最多可分为8个不同的质数相乘的形式，所以可以dfs跑出所有组合利用容斥原理计算出m。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;

vector <ll> prime;
int bit[20];
ll ans,n,n1;

void dfs(int now,int len)
{
```

```
    if(now==len)
    {
        int cnt=0;
        ll x=1;
        for(int i=0;i<len;i++)
        {
            cnt+=bit[i];
            if(bit[i]) x*=prime[i];
        }
        if(x==1) return;
        if(cnt&1) ans+=n1/x;
        else ans-=n1/x;
        return;
    }
    for(int i=0;i<=1;i++)
    {
        bit[now]=i;
        dfs(now+1,len);
        bit[now]=0;
    }
    return;
}

int main()
{
    cin>>n;
    n1=n;
    for(ll i=2;i*i<=n;i++)
    {
        if(n%i==0)
        {
            prime.push_back(i);
            while(n%i==0) n/=i;
        }
        if(n==1) break;
    }
    if(n!=1) prime.push_back(n);
    int len=prime.size();
    dfs(0,len);
    cout<<n1-ans<<"\n";
    return 0;
}
```

7-3

开一个二维数组ans，$ans[i][0]$表示岗位i的得票最多同学的学号，$ans[i][1]$表示岗位i的得票最多同学的票数，根据输入更新ans数组。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;

```

```cpp
int ans[20][5];

int main()
{
    int n,m; cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        int c,t; cin>>c>>t;
        if(ans[c][1]<t)
        {
            ans[c][0]=i;
            ans[c][1]=t;
        }
    }
    for(int i=1;i<=m;i++) cout<<ans[i][0]<<" \n"[i==m];
    return 0;
}
```

7-4

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;


int main()
{
    int n; cin>>n;
    for(int i=0;i<n;i++)
    {
        int x,y,z; cin>>x>>y>>z;
        if(x*x+y*y+z*z==3*x*y*z) cout<<"Yes\n";
        else cout<<"No\n";
    }
    return 0;
}
```

7-5

根据题意判断即可

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;


int main()
```

```cpp
{
    int t; cin>>t;
    while(t--)
    {
        int n,h,m;
        string s;
        cin>>n>>s;
        scanf("%d:%d",&h,&m);
        if(n>=18)
        {
            cout<<"Yes\n";
            continue;
        }
        if((s=="Fri"||s=="Sat"||s=="Sun")&&h==20) cout<<"Yes\n";
        else cout<<"No\n";
    }
    return 0;
}
```

7-6

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;

int days[2][35]={{0,31,28,31,30,31,30,31,31,30,31,30,31},
{0,31,29,31,30,31,30,31,31,30,31,30,31}};

int main()
{
    int y,m,d,ans=0,rn=0;
    scanf("%d/%d/%d",&y,&m,&d);
    if((y%100==0&&y%400==0)||(y%100!=0&&y%4==0)) rn=1;
    for(int i=1;i<m;i++) ans+=days[rn][i];
    ans+=d;
    cout<<ans<<"\n";
    return 0;
}
```

7-7

可知，只有水和岩浆相邻，或者水和岩浆中间只隔了空气，才可能生成新的黑曜石。因此答案为 一开始的黑曜石数 + 按高度排序后水和岩浆相邻的个数。

```cpp
#include <bits/stdc++.h>

#define DEBUG

using namespace std;
```

```cpp
namespace hjt {
template<typename A,typename B>
std::ostream &operator<<(std::ostream &o,const std::pair<A,B> &x){
  return o<<'('<<x.first<<','<<x.second<<')';
}
#define repeat(i,a,b) for(int i=(a),_=(b);i<_;i++)
#define repeat_back(i,a,b) for(int i=(b)-1,_=(a);i>=_;i--)
#define qwq [&]{cerr<<"qwq"<<endl;}()
#define orz(x) [&]{cerr<<#x": "<<x<<endl;}()
#define orzarr(a,n) [&]{cerr<<#a": "; repeat(__,0,n)cerr<<(a)[__]<<" ";
cerr<<endl;}()
#define orzeach(a) [&]{cerr<<#a": "; for(auto __:a)cerr<<__<<" ";
cerr<<endl;}()
#define pause [&]{system("pause");}()
} using namespace hjt;

typedef long long ll;
typedef pair<int, int> pii;
const double eps = 1e-7;
const double PI = acos(-1);
const int MOD = 998244353; // 1e9+7;
const int INF = 0x3f3f3f3f;
// const ll INF = 1e18;
const int N = 1e5+7;

int n;
pii a[N];

inline void solve() {
  cin >> n;
  int ans = 0;
  for (int i = 1; i <= n; ++i) {
    cin >> a[i].second >> a[i].first;
    ans += a[i].second == 3;
  }
  sort(a+1, a+n+1);
  for (int i = 1; i < n; ++i) {
    ans += (a[i].second^a[i+1].second) == 3;
  }
  cout << ans << '\n';
}

signed main() {
#ifdef ONLINE_JUDGE
  ios::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
#endif
  int T = 1;
  cin >> T; // scanf("%d", &T);
  for (int t = 1; t <= T; ++t) {
    solve();
  }
  return 0;
}
```

7-8

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;


int main()
{
    int n,now=1,ans=0; cin>>n;
    for(int i=1;i<=n;i++)
    {
        now*=i;
        ans+=now;
    }
    cout<<ans<<"\n";
    return 0;
}
```

7-9

手动建图，然后枚举所有连通块

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 17;
constexpr int x[] = {1,1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,5};
constexpr int y[] = {1,2,4,5,2,3,4,2,3,4,2,3,4,1,2,4,5};
constexpr int dir[] = {0,-1,0,1,0};

int T, n, vis, cas;
int a[N], mp[7][7];
long long sum;

constexpr void init() {
  memset(mp, -1, sizeof mp);
  for (int i = 0, k = 0; i < N; ++i) {
    mp[x[i]][y[i]] = k++;
  }
}

void dfs(int x, int y) {
  int id = mp[x][y];
  if (id == -1 || (cas>>id)%2 == 0 || (vis>>id)%2) return;
  sum += a[id];
  vis |= 1<<id;
  for (int d = 0; d < 4; ++d) {
    dfs(x+dir[d], y+dir[d+1]);
  }
}

signed main() {
  init();
  cin >> T;
```

```
  while (T--) {
    for (int i = 0; i < N; ++i) {
      cin >> a[i];
      a[i] = a[i]/6+1;
    }
    cin >> n;
    int ans = 0;
    for (cas = 0; cas < 1<<17; ++cas) {
      if ((cas>>13)%2 == 0) continue;
      sum = vis = 0;
      dfs(x[13], y[13]);
      if (vis == cas && sum <= n) {
        ans = max(ans, __builtin_popcount(cas));
      }
    }
    cout << ans << '\n';
  }
  return 0;
}

```

7-10

本题是一个模拟题，按题意实现即可，复杂度为 O(n2)。需要注意以下几个点：

1. 出现多只与狂热者距离相同的异虫时，需要选取最早出现的那只。
2. 考虑狂热者距离时不能考虑已经死亡（离开战场）的异虫。
3. 距离平方的极限数据可达 4×1016，使用 double 计算有可能会产生浮点精度误差。但是注意到距离之间只需要比较大小，因而可将所有距离平方，在长整型范围内进行运算，避免精度误差。

```
#include <bits/stdc++.h>

const int N = 2010;
using ll = long long;
inline ll sqr(ll x){return x * x;}

int x[N], y[N];
int type[N], atk[N], h[N], r[N];
bool leave[N];

ll dis(int pos1, int pos2){
    return sqr(x[pos1] - x[pos2]) + sqr(y[pos1] - y[pos2]);
}

int main(){
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; ++ i){
        scanf("%d", &type[i]);
        if (type[i] == 1){
            scanf("%d%d%d", &x[i], &y[i], &h[i]);
        }
        else{
            scanf("%d%d%d%d", &x[i], &y[i], &atk[i], &r[i]);

```

```
            int pos = -1;
            ll min_dis = LLONG_MAX;
            for (int j = 0; j < i; ++ j){
                if (type[j] == 1 && !leave[j]){
                    ll value = dis(i, j);
                    if (min_dis > value){
                        min_dis = value;
                        pos = j;
                    }
                }
            }
            if (pos >= 0){
                x[i] = x[pos], y[i] = y[pos];
            }
            for (int j = 0; j < i; ++ j){
                if (type[j] == 1 && !leave[j]){
                    if (dis(i, j) <= sqr(r[i])){
                        h[j] -= 3 * atk[i];
                        if (h[j] <= 0){
                            leave[j] = true;
                        }
                        else{
                            leave[i] = true;
                        }
                    }
                }
            }
        }
    }
    for (int i = 0; i < n; ++ i){
        puts(leave[i] ? "No" : "Yes");
    }
    return 0;
}

```