## 比较大小：

这题就不上代码了，使用sort()的代码量最小。

## 一帮一：

思路：i从前面， j,k从后面开始进行遍历。当i碰到的是0性别，j往回找第一个1性别。当i碰到的是1性别，k往回找第一个0性别。

```
#include<stack>
#include<iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    pair<int, string> stu[50];
    for(int i=0; i<n; i++) cin >> stu[i].first >> stu[i].second;
    int j = n-1, k = n-1;
    for(int i=0; i<n/2; i++)
    {
        cout << stu[i].second;
        if(stu[i].first==0)
        {
            while(stu[j].first==0) j--;
            cout << " " << stu[j].second << endl;
            j--;
        }
        else
        {
            while(stu[k].first==1) k--;
            cout << " " << stu[k].second << endl;
            k--;
        }
    }
}
```

## 考试座位号：

思路：因为要查询的是 试机座位号码，要保存的是相应 准考证号 和 正式座位号，所以把 试机号 作为下标 ， 准考证号和正式座位号作为该下标对应的 两个数据。

```
#include<iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    string a[1001][2];
    for(int i=0; i<n; i++)
```

```
    {
        string s;
        int x;
        cin >> s >> x;
        a[x][0] = s;
        cin >> a[x][1];
    }
    cin >> n;
    for(int i=0; i<n; i++)
    {
        int x;
        cin >> x;
        cout << a[x][0] << " " << a[x][1] << endl;
    }
    return 0;
}
```

## 删除重复字符:

又是一道 数据 作为 数组下标 的题目，这个方法超好使

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

int main()
{
    string s;
    getline(cin, s);
    sort(s.begin(), s.end());
    int a[256] = {0};
    for(char x:s)
    {
        if(a[x]==0)
        {
            cout << x;
            a[x] = 1;
        }
    }
    return 0;
}
```

## 最长的括号子串:

技巧在于：下标入栈 而不是 符号入栈。

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

int main()
{
    string s;
    cin >> s;
    stack<int> st;
    int res = 0;
    for(int i=0; s[i]; i++)
    {
```

```
        if(s[i]==')' && !st.empty() && s[st.top()]=='(')
        {
            st.pop();
            if(st.empty()) res = i+1;
            else res = max(res, i-st.top());
        }
        else st.push(i);
    }
    cout << res;
    return 0;
}
```

## 家庭房产：

在读入数据时使用 并查集 不断合并家庭。然后以 人均面积+编号排序 排序，最后输出。不算难，就是有点烦，要小心。

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

struct node
{
    int id=-1, num;
    float house=0, area=0;
};

int pre[10000];

int root(int x)
{
    while(pre[x]!=x) x = pre[x];
    return x;
}

void un(int x, int y)
{
    int rootx = root(x);
    int rooty = root(y);
    if(rootx<rooty) pre[rooty] = rootx;
    else pre[rootx] = rooty;
}

bool cmp(node x, node y)
{
    if(x.area/x.num==y.area/y.num) return x.id<y.id;
    else return x.area/x.num > y.area/y.num;
}

int main()
{
    for(int i=0; i<10000; i++) pre[i] = i;
    node arr[10000];
    bool exist[10000] = {0};
    int n;
    cin >> n;
```

```
    for(int i=0; i<n; i++)
    {
        int id, fa, ma;
        cin >> id >> fa >> ma;
        arr[id].id = id;
        exist[id] = 1;
        if(fa!=-1) exist[fa]=1, un(id, fa);
        if(ma!=-1) exist[ma]=1, un(id, ma);
        int k, x;
        cin >> k;
        while(k--)
        {
            cin >> x;
            un(id, x);
            exist[x]=1;
        }
        cin >> arr[id].house >> arr[id].area;
    }
    node tmp[10000];
    for(int i=0; i<10000; i++)
    {
        if(exist[i])//该编号存在
        {
            int f = root(i);//该编号所在家庭的最小编号
            tmp[f].id = f;
            tmp[f].num++;
            if(arr[i].id!=-1)//该编号有房产
            {
                tmp[f].house += arr[i].house;
                tmp[f].area += arr[i].area;
            }
        }
    }
    node res[10000];
    int i = 0;
    for(int j=0; j<10000; j++)//把每一个家庭复制到res
        if(tmp[j].id!=-1) res[i++] = tmp[j];
    sort(res, res+i, cmp);
    cout << i << endl;
    for(int j=0; j<i; j++)
        printf("%04d %d %.3f %.3f\n", res[j].id, res[j].num,
res[j].house/res[j].num, res[j].area/res[j].num);
    return 0;
}
```

## 直直直径：

你把它看成图就好办了，无非就是深搜每一条路径，过程中刷新最大值res

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

vector<pair<int, int>> a[200000];
long long res = 0;
int n;
```

```
void dfs(int t, bool vis[], long long len)
{
    if(vis[t]==1) return;
    vis[t] = 1;
    if(len>res) res = len;
    for(auto x:a[t])
        if(!vis[x.first]) dfs(x.first, vis, len+x.second);
}

int main()
{
    cin >> n;
    for(int i=0; i<n-1; i++)
    {
        int x, y, w;
        cin >> x >> y >> w;
        a[x].push_back({y, w});
        a[y].push_back({x, w});
    }
    for(int i=1; i<n; i++)
        if(a[i].size()==1)
        {
            bool vis[200000] = {0};
            dfs(i, vis, 0);
        }
    cout << res;
    return 0;
}
```

## 储水问题

1. left，right先定位到左边、右边的第一个围栏（能把水围住那栏）下标。
2. x=left的栏高, y=right的栏高
3. 如果 本次 x<y，那就从left+1（下标）开始，顺序计算接下来的每一栏比x低多少并累计，直到某栏高于x。期间left++
如果 本次 x>=y, 那就从right-1（下标）开始，顺序计算接下来的每一栏比y低多少并累计，直到某栏高于y。期间right--
4. 回到第二点，直到left>=right

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

int a[1000000];

int main()
{
    int n;
    cin >> n;
    for(int i=0; i<n; i++) cin >> a[i];
    int left, right;
    //找左边围栏下标
    for(left=0; left<n-1 && a[left]<=a[left+1]; left++);
    //找右边围栏下标
    for(right=n-1; right>0 && a[right]<=a[right-1]; right--);
    long long res = 0;
    int x, y;
```

```
    while(1)
    {
        if(left>=right) break;
        x = a[left];
        y = a[right];
        if(x<y)
            while(left<right && a[++left]<=x) res += x-a[left];
        else
            while(left<right && a[--right]<=y) res += y-a[right];
    }
    cout << res;
}
```

## 德才论：

我觉得这个挺简单的，小心点就行

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;

struct person
{
    string num;
    int de;
    int cai;
};

bool cmp(person x, person y)
{
    if(x.de+x.cai != y.de+y.cai) return x.de+x.cai > y.de+y.cai;
    else if(x.de != y.de) return x.de > y.de;
    else return x.num < y.num;
}

void output(vector<person> x)
{
    for(auto t:x)
        cout << t.num << " " << t.de << " " << t.cai << endl;
}

int main()
{
    int n, l, h;
    cin >> n >> l >> h;
    string num;
    int x, y;
    vector <person> c1, c2, c3, c4;
    for(int i=0; i<n; i++)
    {
        cin >> num >> x >> y;
        if(x>=h && y>=h) c1.push_back({num, x, y});
        else if(x>=h && y>=l) c2.push_back({num, x, y});
        else if(x>=l && y>=l && x>=y) c3.push_back({num, x, y});
        else if(x>=l && y>=l) c4.push_back({num, x, y});
    }
```

```
    sort(c1.begin(), c1.end(), cmp);
    sort(c2.begin(), c2.end(), cmp);
    sort(c3.begin(), c3.end(), cmp);
    sort(c4.begin(), c4.end(), cmp);
    cout << c1.size()+c2.size()+c3.size()+c4.size() << endl;
    output(c1);
    output(c2);
    output(c3);
    output(c4);
    return 0;
}
```

## 银行排队问题之单窗口“夹塞”版：

fri存储圈子，cus存储每位顾客
顺序遍历cus，处理当前顾客以及他的已到达的朋友
注意：1. 当前要处理的顾客，可能是上一位结束后一段时间才到达，要注意累计时间的计算； 2. 往后找加塞时要判断他们是否朋友 && 朋友是否已到达；

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

map<string, pair<int, int>> cus;//姓名:<到达时间，时长>
map<string, int> fri;//姓名：圈子
string a[10000];

int main()
{
    int n, m;
    cin >> n >> m;
    for(int i=0; i<m; i++)
    {
        int k;
        cin >> k;
        while(k--)
        {
            string s;
            cin >> s;
            fri[s] = i;
        }
    }
    for(int i=0; i<n; i++)
    {
        int x, y;
        cin >> a[i] >> x >> y;
        cus[a[i]] = {x, min(60, y)};
    }
    int now = 0;
    int sum = 0;
    for(int i=0; i<n; i++)
    {
        auto tmp = cus[a[i]];
        if(tmp.first==-1)//已服务
```

```
            continue;
        cout << a[i] << endl;
        sum += max(0, now-tmp.first);
        now = max(now, tmp.first) + tmp.second;
        //cout << "sum: " << sum << endl;
        for(int j=i+1; j<n; j++)
        {
            //cout << "jia: " << a[j] << endl;
            auto tmp = cus[a[j]];
            auto x = fri.find(a[i]), y = fri.find(a[j]);
            if(x!=fri.end() && y!=fri.end())
                if(tmp.first!=-1 && x->second==y->second && tmp.first<=now)
                {//加塞
                    cout << a[j] << endl;
                    cus[a[j]].first = -1;
                    sum += (now-tmp.first);
                    now += tmp.second;
                    //cout << "sum: " << sum << endl;
                }
        }
    }
    printf("%.1f", 1.0*sum/n);
    return 0;
}
```