GMCPC_solution

A. Alice的秘密

```
#include<stdio.h>
#include<string.h>

int isDate(int year, int month, int day){
        if(year>2020 || year<1900) return 0;
        if(month>12 || month<1) return 0;
        if(month==1||month==3||month==5||month==7||month==8||month==10||month==12){
                return (day>=1&&day<=31);
        }
        if(month==4||month==6||month==9||month==11) {
                return (day>=1&&day<=30);
        }
        if((year%4==0&&year%100!=0)||(year%400==0)){
                return (day>=1&&day<=29);
        }
        return day>=1&&day<=28;
}

int getK(char *date){
        if(strlen(date)!=8) return 0;
        int d[8];
        for(int i=0; i<8; i++){
                if(date[i]>'9'||date[i]<'0') return 0;
                d[i] = date[i]-'0';
        }
        int year = d[0]*1000 + d[1]*100 + d[2]*10 + d[3];
        int month = d[4]*10 + d[5];
        int day = d[6]*10 + d[7];
        if(!isDate(year, month, day)) return 0;
        int k=0;
        for(int i=0; i<8; i++){
                k += d[i];
        }
```

```
        while(k>9){
                int sum = k;
                k = 0;
                while(sum>0){
                        k+=sum%10;
                        sum/=10;
                }
        }
        return k;
}

int isMsg(char *msg){
        int i=0;
        while(msg[i]!='\0'){
                if(!((msg[i]==' ') ||
                        (msg[i]<='z'&&msg[i]>='a') || (msg[i]<='Z'&&msg[i]>='A')))
                        return 0;
                i++;
        }
        return 1;
}

void printMsg(char *msg, int k){
        int i=0;
        while(msg[i]!='\0'){
                if(msg[i]==' ') printf("#");
                else{
                        if(msg[i]>='a'&&msg[i]<='z')
                                printf("%c", (msg[i]+k-'a')%26+'a');
                        else printf("%c", (msg[i]+k-'A')%26+'A');
                }
                i++;
        }
        printf("\n");
}
int main(){
        char date[10];
        char msg[132];
        while(gets(date)!=NULL){
                gets(msg)!=NULL;
                int k = getK(date);
                if(k==0 || !isMsg(msg)) printf("none\n");
                else{
                        printMsg(msg, k);

                }
        }
}
```

B. 今天星期几

```
#include<stdio.h>

int isLeap(int year){
        return (year%4==0&&year%100!=0)||(year%400==0);
}

int daysInMonth(int year, int month){
        switch(month){
                case 1: return 31;
                case 2: if(isLeap(year)) return 29;
                                else return 28;
                case 3: return 31;
                case 4: return 30;
                case 5: return 31;
                case 6: return 30;
                case 7: return 31;
                case 8: return 31;
                case 9: return 30;
                case 10: return 31;
                case 11: return 30;
                case 12: return 31;
        }
}

int main(){
        int year, month, day, sum=0;
        while(~scanf("%d %d %d", &year, &month, &day)){
                sum = 0;
                for(int y=2000; y<year; y++){
                if(isLeap(y)) sum+=366;
                else sum+=365;
                }
                for(int m=1; m<month; m++){
                        sum += daysInMonth(year, m);
                }
                for(int d=1; d<day; d++){
                        sum++;
                }
                printf("%d %d\n", sum/4+1, sum%4+1);
        }
}
```

C. 小明的英文作业

```
#include<stdio.h>
#include<string.h>
#include<string>
#include<map>
using namespace std;

int main(){
        char *cp, line[2200], word[24];
        while(gets(line)){
                int sum = 0;
                if(line[0] == '#') break;
                else{
                        map<string, int> m;
                        cp = line;
                        while(*cp!='\0'){
                                int i = 0;
                                while(*cp !=' '&& *cp!='\0'){
                                        word[i] = *cp;
                                        i++;
                                        cp++;
                                }
                                if (*cp==' ') cp++;
                                word[i] = '\0';
                                m[word]++;
                                sum++;
                        }
                        int count = 0;
                        map<string,int> ::iterator it;
                        for(it=m.begin(); it!=m.end(); it++){
                                if(it->second==1) count++;
                        }
                        if(count*2>=sum) printf("yes\n");
                        else printf("no\n");
                }
        }
}
```

D. 分三排

```
import java.util.ArrayList;
import java.util.Scanner;

public class Three {
	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		while(input.hasNext()) {
			ArrayList<Integer> list1 = new ArrayList<Integer>();
			ArrayList<Integer> list2 = new ArrayList<Integer>();
			ArrayList<Integer> list3 = new ArrayList<Integer>();
			int count1 = 0,count2 = 0,count3 = 0;
			list1.clear();
			list2.clear();
			list3.clear();
			int total;
			total = input.nextInt();
			if(total == 1) {
				System.out.println(input.nextInt());
			}else if(total == 2) {
				System.out.println(input.nextInt());
				System.out.println(input.nextInt());
			}else {
				for(int i = 1;i<=total;i++) {
					if(i % 3 == 1) {
						list1.add(input.nextInt());
						count1++;
					}else if(i % 3 == 2) {
						list2.add(input.nextInt());
						count2++;
					}else {
						list3.add(input.nextInt());
						count3++;
					}
				}
				for(int i=0;i<count1-1;i++) {
					System.out.print(list1.get(i) + " ");
				}
				System.out.println(list1.get(count1-1));

				for(int i=0;i<count2-1;i++) {
					System.out.print(list2.get(i) + " ");
				}
				System.out.println(list2.get(count2-1));

				for(int i=0;i<count3-1;i++) {
					System.out.print(list3.get(i) + " ");
				}
				System.out.println(list3.get(count3-1));
			}
		}
		input.close();
```

```
        }
}
```

F.序列计数

本题是根据某个老题的无聊改编。

题目的式子是多重集合的排列数，即

$$\frac{K!}{k_1!k_2!\dots k_n!}=C_K^{k_1}C_{K-k_1}^{k_2}C_{K-k_1-k_2}^{k_3}\cdots C_{K-k_1-k_2\cdots-k_{n-1}}^{k_n}$$

$P=13$，是个质数，考虑用Lucas定理，那么$C_n^m \bmod P=0$当且仅当P进制下m的某一位的数值大于n对应位的数值，相反$C_n^m \bmod P\neq 0$当且仅当P进制下m的每一位的数值都小于等于n对应位的数值。

计算反面，统计原式不是P的倍数的情况。那么$C_K^{k_1},C_{K-k_1}^{k_2},\dots$模$P$都不为$0$，根据Lucas定理再进行推导会发现，将$k_1,k_2,\dots,k_n$分别写成$P$进制，那么对于每一进制位都需要满足，$n$个数字之和不超过$P$。因为如果超过$P$则用Lucas定理计算比如$C_{k_1+Sum}^{k_1}$这个组合数的时候，那一位就会发生进位，进位则导致结果模P为0。所以这里用数位dp求解即可，这部分详细过程略。最后用总数$\prod_{i=1}^{n}(a_i+1)$减去就是答案。

```
#include<bits/stdc++.h>
using namespace std;
const int MOD = 1e9+7;
const int prime = 13;
int n;
int digit[10][10];
int dp[10][10][14][1<<10];
void Add(int &a,int b){
        a += b;
        if(a >= MOD)a -= MOD;
}
```

```
int dfs(int row,int pos,int dig_sum,int limit){
        if(row == n){
                row = 0;
                pos--;
                dig_sum = 0;
                if(pos == -1)return 1;
        }
        if(dp[row][pos][dig_sum][limit] != -1)
                return dp[row][pos][dig_sum][limit];
        int ans = 0;
        int end = ((limit>>row)&1)?digit[row][pos]:prime-1;
        for(int i = 0;i <= end;i++){
                if(i + dig_sum >= prime)break;
                Add(ans,dfs(row+1,pos,dig_sum+i,i == end? limit : (limit & (~(1<<row)))));
        }
        dp[row][pos][dig_sum][limit] = ans;
        return ans;
}
int x[10];
int calc(){
        memset(digit,0,sizeof(digit));
        memset(dp,-1,sizeof(dp));
        int cnt = 0;
        for(int i = 0;i < n;i++){
                int pos = 0;
                do{
                        digit[i][pos++] = x[i]%prime;
                        x[i] /= prime;
                }
                while(x[i]);
                cnt = max(cnt,pos);
        }
        return dfs(0,cnt-1,0,(1<<n)-1);
}

int main(){
    int T;
        int iCase = 0;
        scanf("%d",&T);
        while(T--){
                iCase++;
                scanf("%d",&n);
                long long tot = 1;
                for(int i = 0;i < n;i++){
                        scanf("%d",&x[i]);
                        tot = tot*(x[i]+1)%MOD;
                }
                tot -= calc();
                tot = (tot%MOD+MOD)%MOD;
                printf("Case #%d: %d\n",iCase,(int)tot);
        }
```

```
    return 0;
}
```

G. 强迫症

```
#include <stdio.h>
#include <map>
using namespace std;

int main(){
    map<long long int, int> mp;
    int n;
    long long int index;
    scanf("%d", &n);
    for(int i=0; i<n; i++){
        scanf("%lld", &index);
        mp[index]++;
        }
        map<long long int, int>::iterator iter;
        iter = mp.begin();
        while(iter != mp.end()) {
        if(iter->second>n/2){
                printf("%lld", iter->first);
                break;
                }
        iter++;
    }
        return 0;
}
```

H. 新型冠状爱情病毒

区间第x大和第1大的异或和。
因为数据不重复，先将数据构建成带下标的双向链表，将所有数排序后，从小到大遍历每个下标。
往左扫x个，往右扫x个，然后尺取x个数，就能知道当前区间第k大是谁。
扫的时候顺带维护一个前缀最大值和后缀最大值，就能知道第1大的值是谁。
转移的时候能很容易算到有多少区间第k大是自己，贡献也自然能算出来了。
由于是带下标的双向链表维护，读的时候是O(1)，删除的时候是O(1)，左右扫描x个，只是x常数级别。
总体来说时间复杂度是O(Tnx)，x比log(n)大，就忽略了排序的时间复杂度了。
顺带说一句，标程跑了300ms，把带log n常数的查询，比如线段树之类的卡掉了。所以有些选手1s多的程序没跑过去。可以考虑用ST表或者前后缀记录最大值的方式通过本题。

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

#define lson l, mid, root << 1
#define rson mid + 1, r, root << 1 | 1
#define MAXN 100005
#define ll long long

using namespace std;

const ll INF = 0x3f3f3f;

ll nums[MAXN];
ll F[MAXN][20];
int pre[MAXN], nxt[MAXN], pos[MAXN], idx[MAXN];
ll rgt[105], lft[105];
ll rgtMax[105], lftMax[105];
int n, k;

struct Node {
    ll val;
    int idx;
} nod[MAXN];

int cmp(Node a, Node b) {
    if (a.val == b.val) {
        return a.idx < b.idx;
    }
    return a.val < b.val;
}

void remove(int p) {
    int tmp1 = pre[p];
    int tmp2 = nxt[p];
    pre[tmp2] = tmp1;
    nxt[tmp1] = tmp2;
}

ll solve() {
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        pre[i] = i - 1;
        nxt[i] = i + 1;
    }
    nums[0] = nums[n + 1] = n + 1;
    for (int i = 1; i <= n; i++) {
        int p = pos[i];
        int l = 0, r = 0;
        lftMax[l] = rgtMax[r] = nums[p];
        lft[l++] = p;
        rgt[r++] = p;
        for (int j = 0, q = pre[p]; j < k; j++, q = pre[q]) {
```

```
            lftMax[l] = max(nums[q], lftMax[l - 1]);
            lft[l++] = q;
            if (q == 0) {
                break;
            }
        }
        for (int j = 0, q = nxt[p]; j < k; j++, q = nxt[q]) {
            rgtMax[r] = max(nums[q], rgtMax[r - 1]);
            rgt[r++] = q;
            if (q == n + 1) {
                break;
            }
        }
        int b = k - (l - 1) + 1;
        int a = l - 2;
        while (a >= 0 && b < r) {
            ll kv = max(lftMax[a], rgtMax[b - 1]);
            ll cal = 1LL * (lft[a] - lft[a + 1]) * (rgt[b] - rgt[b - 1]) * (kv ^ nums[p]);
            ans += cal;
            a--, b++;
        }
        remove(p);
    }
    return ans;
}

int handler() {
    int T, cas = 1;
    scanf("%d", &T);
    while (T--) {
        scanf("%d %d", &n, &k);
        for (int i = 1; i <= n; i++) {
            scanf("%d", &nums[i]);
            nod[i].val = nums[i];
            nod[i].idx = i;
        }
        sort(nod + 1, nod + n + 1, cmp);
        for (int i = 1; i <= n; i++) {
            pos[i] = nod[i].idx;
            idx[nod[i].idx] = i;
        }
        printf("Case #%d: %lld\n", cas++, solve());
    }
    return 0;
}

int main() {
    handler();
}
```

I. 期末表彰

```
#include <stdio.h>

void sort(int sum[],int c[],int gs[],int en[],int num[],int n){
        int i,j,t,flag=0;
        for(i=1;i<n;i++){
                flag=0;
                for(j=0;j<n-i;j++){
                        if(sum[j]<sum[j+1]){
                                flag=1;
                        }else if(sum[j]==sum[j+1]){
                                if(c[j]<c[j+1]){
                                        flag=1;
                                }
                        }

                        if(flag==1){
                                flag=0;
                                t=sum[j];
                                sum[j]=sum[j+1];
                                sum[j+1]=t;
                                t=gs[j];
                                gs[j]=gs[j+1];
                                gs[j+1]=t;
                                t=c[j];
                                c[j]=c[j+1];
                                c[j+1]=t;
                                t=en[j];
                                en[j]=en[j+1];
                                en[j+1]=t;
                                t=num[j];
                                num[j]=num[j+1];
                                num[j+1]=t;
                        }
                }
        }
}
```

```
int main()
{
    int i,n,c[100],gs[100],en[100],sum[100],num[100];
    while(~scanf("%d",&n)){
        for(i=0;i<n;i++){
                num[i]=i+1;
                scanf("%d%d%d",&c[i],&gs[i],&en[i]);
                sum[i]=gs[i]+c[i]+en[i];
        }
        sort(sum,c,gs,en,num,n);
        for(i=0;i<5;i++)
           printf("%d %d %d %d\n",num[i],c[i],gs[i],en[i]);
        }
    return 0;
}
```

K. 项目管理

```
#include <stdio.h>
int partition(int bz[],int work[],int left,int right){
        int b,w,temp;
        w=work[left];
        b=bz[left];
        while(left<right){
                while(work[right]<=w && left <right)
                        right--;
                if(right!=left){
                        work[left]=work[right];
                        bz[left]=bz[right];
                        left++;         }
                while(work[left]>=w&&left<right)
                        left++;
                if(right!=left){
                        work[right]=work[left];
                        bz[right]=bz[left];
                        right--;
                }
        }
        work[left]=w;
        bz[left]=b;
        return left;
}
```

```
void quicksort(int bz[],int work[],int low,int high){
        int p;
        p=partition(bz,work,low,high);
        if(low<p)
                quicksort(bz,work,low,p-1);
        if(high>p)
                quicksort(bz,work,p+1,high);
}
int main(){
        int i,n;
        int bz_t[10000],work_t[10000];


        while(~scanf("%d",&n)){
                int bz_tot=0,work_tot=0;
                for(i=0;i<n;i++){
                        scanf("%d%d",&bz_t[i],&work_t[i]);
                }
                quicksort(bz_t,work_t,0,n-1);
                for(i=0;i<n;i++){
                        bz_tot+=bz_t[i];
                        work_tot=work_tot>bz_tot+work_t[i]?work_tot:bz_tot+work_t[i];
                }
                printf("Project %d: %d\n",n,work_tot);

        }
        return 0;
}
```

L. 捕鱼达人

题意其实很明显就是个带权值的上升序列。不管哪种计算模数的方式去计算负数，都会发现负数的贡献都是0，没必要选，处理数据的时候就应该将负数去掉了，没必要纠结负数取模的结果。
可以通过树状数组或者线段树的方式维护一个第x大到无穷大的权值。最后只需要知道棵树里最大的值即可。
当然，因为题目里要求的是非递减序列，不会维护树方式的同学，完全可以将1个5费权值的鱼拆开，比如颜值50000的拆成5个10000的鱼，然后去跑LIS。这里要注意由于数据量级n是10的4次方，LIS需要跑nlogn的。

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

#define MAXN 100005
#define ll long long

using namespace std;

int nums[MAXN];
int st[MAXN];

int getLIS(int len) {
    if (len == 0) {
        return 0;
    }
    int stLen = 0;
    st[stLen++] = nums[0];
    for (int i = 1; i < len; i++) {
        int idx = upper_bound(st, st + stLen, nums[i]) - st;
        if (idx == stLen) {
            st[stLen++] = nums[i];
        } else {
            st[idx] = nums[i];
        }
    }
    return stLen;
}

void revertNums(int len) {
    for (int i = 0; i < len / 2; i++) {
        swap(nums[i], nums[len - i - 1]);
    }
}

int main() {
    int T, cas = 1;
    int n, tmp;
    scanf("%d", &T);
    while (T--) {
        scanf("%d", &n);
        int len = 0;
        for (int i = 0; i < n; i++) {
            scanf("%d", &tmp);
            if (tmp <= 0) {
                continue;
            }
            for (int j = 0; j < tmp / 10000; j++) {
                nums[len++] = tmp % 10000;
            }
        }
        int ans = getLIS(len);
        revertNums(len);
```

```
        ans = max(getLIS(len), ans);
        printf("Case #%d: %d\n", cas++, ans);
    }
}
```

M. 排除危险

```
#include <stdio.h>
#define maxn 100000
int pa[maxn];

int findset(int x) { return pa[x] != x ? pa[x] = findset(pa[x]) : x; }

int main() {
  int i,x, y;
  while(scanf("%d", &x) == 1) {
    for(i = 0; i <= maxn; i++) pa[i] = i;
    int refusals = 0;
    while(x != -1) {
      scanf("%d", &y);
      x = findset(x); y = findset(y);
      if(x == y) ++refusals;
      else pa[x] = y;
      scanf("%d", &x);
    }
    printf("%d\n", refusals);
  }
  return 0;
}
```

N. 图像编码问题

```
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Random;
```

```
public class Main{
	public static class Pixel{
		int r;
		int c;
		int v;
		int m;

		public Pixel(int r,int c,int v) {
			this.r=r;
			this.c=c;
			this.v=v;
			morton();
		}

		private void morton() {
			String br=Integer.toBinaryString(r);
			String bc=Integer.toBinaryString(c);

			StringBuilder sb=new StringBuilder();
			if(br.length()!=bc.length()) {
				if(br.length()>bc.length()) {
					for(int i=0;i<br.length()-bc.length();i++)
						sb.append("0");
					sb.append(bc);
					bc=sb.toString();
				}
				else {
					for(int i=0;i<bc.length()-br.length();i++)
						sb.append("0");
					sb.append(br);
					br=sb.toString();
				}
			}

			sb.delete(0,sb.length());
			for(int i=0;i<br.length();i++) {
				sb.append(br.charAt(i));
				sb.append(bc.charAt(i));
			}
			m=Integer.parseInt(sb.toString(),2);
		}

		public int getV() {
			return this.v;
		}

		public int getMorton(){
			return this.m;
		}
	}
```

```
        public static void main(String[] args) {
                  Scanner scan = new Scanner(System.in);
                  int n=Integer.parseInt(scan.nextLine());
                 String[] ss=new String[n];
                 int m=0;
                 while(scan.hasNext()){
                     ss[m++]=scan.nextLine();
                 }

                List<Pixel> pixels=new ArrayList<Pixel>();
                for(int i=0;i<n;i++) {
                    String[] line=ss[i].split(" ");
                    for(int k=0;k<line.length;k++) {
                            pixels.add(new Pixel(i, k, Integer.parseInt(line[k])));
                    }
                }
                pixels.sort(Comparator.comparingInt(Pixel::getMorton));

                    StringBuilder sb=new StringBuilder();
                 Pixel pixel=pixels.get(0);
                 int p=pixel.getV();
                 int count=1;
                 for(int i=1;i<n*n;i++) {
               pixel=pixels.get(i);
                           if(p==pixel.getV()) {
                           count++;
                           }
                           else {
                                   sb.append(String.valueOf(p)+","+String.valueOf(count)+" ");
                                   p=pixel.getV();
                                   count=1;
                           }
                 }
                     sb.append(String.valueOf(p)+","+String.valueOf(count)+" ");
                 System.out.print(sb.toString().substring(0,sb.length()-1));
        }
}
```

O. 军训值日生

```
#include<stdio.h>
#include<string.h>
#define MAX 100

int dp[MAX+1][MAX+1];
int sum[MAX+1], input[MAX+1];
int n;
```

```
void energy(){
    for(int r=2;r<=n;r++){
        for(int i=1;i<=n-r+1;i++){
            int j=i+r-1;
            dp[i][j]=dp[i][i]+dp[i+1][j]+(sum[i]-sum[i-1])*(sum[j]-sum[i]);
            for(int k=i+1;k<j;k++){
                int t=dp[i][k]+dp[k+1][j]+(sum[k]-sum[i-1])*(sum[j]-sum[k]);
                if(t<dp[i][j]){
                    dp[i][j]=t;
                }

            }
        }
    }
}
int main(){
        while(~scanf("%d", &n)){
                sum[0] = 0;
                int num;
                for(int i=1; i<=n; i++){
                        scanf("%d", &num);
                        sum[i] = sum[i-1] + num;
                        dp[i][i] = 0;
                }
                energy();
        printf("%d\n", dp[1][n]);
        }
        return 0;
}
```

P. 今天图书馆开了没?

贪心。每天，维护每一间阅览室的最优解。当天一间阅览室的最优解等于前一天除了此间阅览室之外的其他阅览室中的最优解+1；对于不开放的阅览室，直接拿前一天全部阅览室的最优解作为该阅览室的最优解。维护到最后一天即有答案。

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

using namespace std;

int main() {
    int dp[4];
    dp[0] = dp[1] = dp[2] = dp[3] = 0;
    int ma1, ma2;
    ma1 = ma2 = 0;
    int n;
```

```
    scanf("%d", &n);
    int a[4];
    for(int i = 0; i < n; i++) {
        scanf("%d %d %d %d", &a[0], &a[1], &a[2], &a[3]);
        for(int j = 0; j < 4; j++) {
            if(a[j] == 0) {
                dp[j] = ma1;
            } else {
                if(dp[j] == ma1) {
                    dp[j] = ma2 + 1;
                } else {
                    dp[j] = ma1 + 1;
                }
            }
        }
        for(int j = 0; j < 4; j++) {
            if(dp[j] >= ma1) {
                ma2 = ma1;
                ma1 = dp[j];
            }
        }
    }
    printf("%d\n", ma1);
    return 0;
}
```

Q. 小明的体育课

```
#include<stdio.h>
int step(int n){
        if(n==1) return 1;
        if(n==0) return 1;
        return step(n-1) + step(n-2);
}
int main(){
        int n;
        while(~scanf("%d", &n)){
                printf("%d\n", step(n));
        }
}
```

R. 孤独的字符串

考虑计算反面，即长度为n的非孤独的字符串，字符串中任意一个'a'都要与另一个'a'相邻，不存在单独的一个'a'，那么记这样的串的个数为f_n。

考虑在长度为$n-1$的非孤串结尾加上字符'a'或'b'：

- 加上'a'后，字符串可能以ba结尾，不是非孤串，数量f_{n-2}
- 而结尾是baa的串则不会产生，需要额外加进答案，数量f_{n-3}

$$f_n = 2f_{n-1} - f_{n-2} + f_{n-3}$$

或者换一种方式考虑，所有非孤独的字符串都是由更短的非孤串结尾加上b，aa，baaa得到：

$$f_n = f_{n-1} + f_{n-2} + f_{n-4}$$

所以答案就是$2^n - f_n$。

n最大100，推荐使用python提交，用c++的__int128也可以通过。

```
a=[0 for i in range(100+1)]
a[1]=1
a[2]=2
a[3]=4
a[4]=7
for i in range(5,100+1):
        a[i]=a[i-1]+a[i-2]+a[i-4]

t=int(input().strip())
for i in range(t):
    n=int(input().strip())
    print(2**n-a[n])
```

S. 鲍勃的输入法

字典树。维护一棵字典树，进行建树、查询操作即可。

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

using namespace std;
```

```
struct tnode {
    tnode *ch[26];
    bool eflag;
    tnode() {
        memset(this->ch, NULL, sizeof(this->ch));
        this->eflag = false;
    }
};

tnode *root;

void add(char word[]) {
    tnode *curr = root;
    int len = strlen(word);
    for(int i = 0; i < len; i++) {
        if(curr->ch[word[i] - 'a'] == NULL) {
            curr->ch[word[i] - 'a'] = new tnode();
        }
        curr = curr->ch[word[i] - 'a'];
    }
    curr->eflag = true;
}

int wordNum;

void read(char pre[], int len, tnode *curr) {
    if(curr->eflag) {
        pre[len] = '\0';
        printf("%s\n", pre);
        wordNum++;
        if(wordNum == 50) {
            return;
        }
    }
    for(int i = 0; i < 26; i++) {
        if(curr->ch[i] != 0) {
            pre[len] = 'a' + i;
            read(pre, len + 1, curr->ch[i]);
            if(wordNum == 50) {
                return;
            }
        }
    }
}

char iword[55];

int main() {
    root = new tnode();
    int n;
    scanf("%d", &n);
```

```
    if(n > 100000)
                return 0;
    for(int i = 0; i < n; i++) {
        scanf("%s", iword);
        if(strlen(iword) > 50)
                        return 0;
        add(iword);
    }
    scanf("%d", &n);
    if(n > 10000)
                return 0;
    for(int i = 0; i < n; i++) {
        scanf("%s", iword);
        printf("%s\n", iword);
        tnode *curr = root;
        int ilen = strlen(iword);
        if(ilen > 50)
                        return 0;
        for(int j = 0; j < ilen; j++) {
            curr = curr->ch[iword[j] - 'a'];
            if(curr == NULL) {
                break;
            }
        }
        if(curr != NULL) {
            wordNum = 1;
            for(int j = 0; j < 26; j++) {
                if(curr->ch[j] != NULL) {
                    iword[ilen] = 'a' + j;
                    read(iword, ilen + 1, curr->ch[j]);
                    if(wordNum == 50) {
                        break;
                    }
                }
            }
        } else {
            add(iword);
        }
    }
    return 0;
}
```

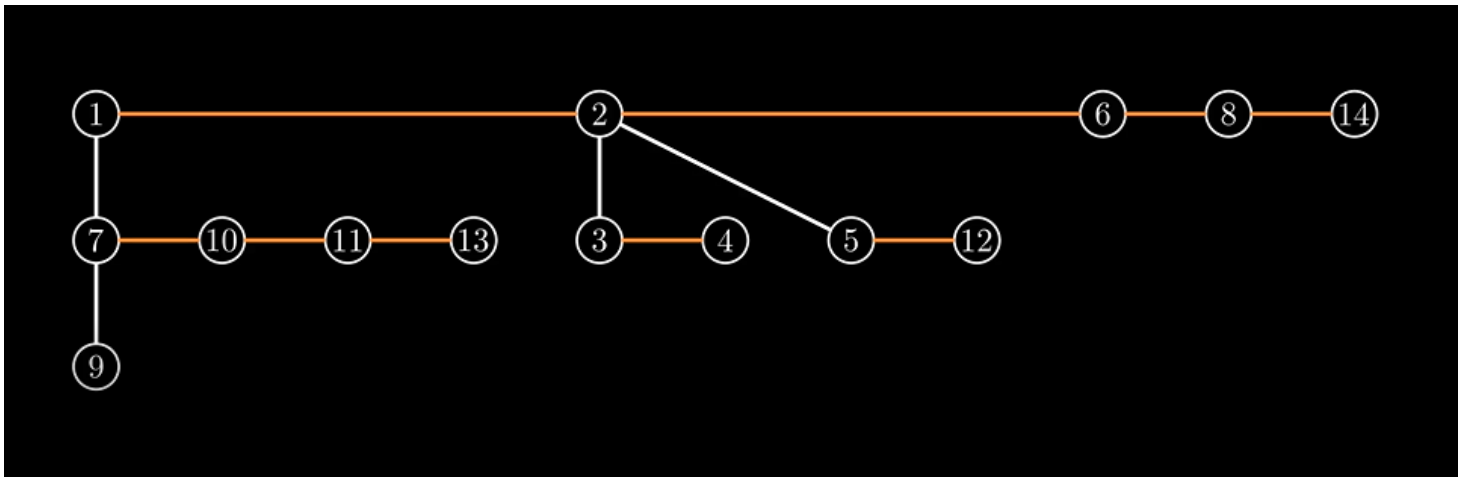
T.一起做课件

题意其实就是可视化一棵树，构造出每个点的坐标。

如果没有面积限制，那么随意定一个根，每个点以深度作为y坐标，每一层横向按照dfs序排列即可。

有面积限制的话，直接不加思考的按层放置面积最坏显然是$O(n^2)$级别的。

考虑树链剖分。首先也是随意定根，放在左上角。每次将重边横向放置，即把含有结点个数最多的子节点放在当前节点的同层，其余子节点放置在下一层，如图所示：



这样树的宽度不超过n，高度不超过$\log n$，总面积$O(n \log n)$。时间复杂度$O(n)$。

这种画树的方法叫Right-Heavy-HV-Tree-Draw，在09年集训队论文《分治算法在树的路径问题中的应用》中也出现过这一算法。

实际上对于本题来说，数据范围与限制较为宽松，欢迎各种奇怪姿势的乱搞。

```
#include<bits/stdc++.h>
using namespace std;
#define N 1010
vector<int> e[N];
int siz[N],son[N],width[N];
int W,H;
int n;
int px[N],py[N];
void dfs(int u,int pre){
    siz[u]=1;
    for(auto &v:e[u]){
        if(v==pre) continue;
        dfs(v,u);
        siz[u]+=siz[v];
        if(son[u]==-1||siz[v]>siz[son[u]])
            son[u]=v;
    }
}
```

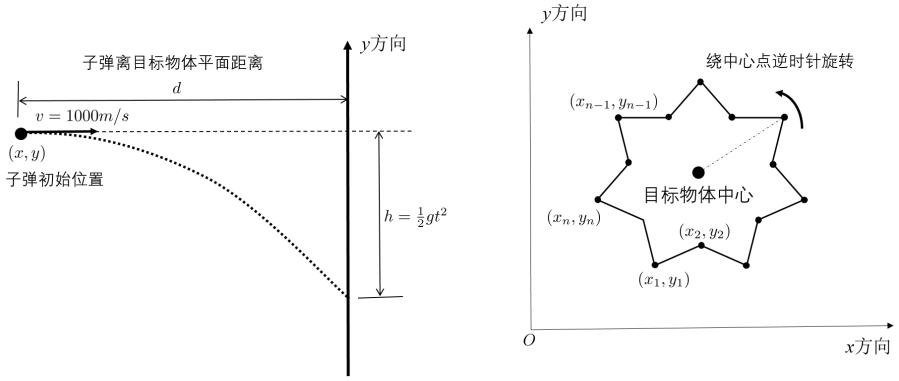
```
void work(int u,int pre,int x,int y){
    W=max(W,y);
    H=max(H,x);
    px[u]=x;py[u]=y;
    if(son[u]==-1) return;
    int sum=0;
    for(auto &v:e[u]){
        if(v==pre || v==son[u]) continue;
        work(v,u,x+1,y+sum);
        sum+=width[v]+1;
    }
    if(sum==0) sum+=1;
    work(son[u],u,x,y+sum);
    width[u]=W-y;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<n;++i){
        int u,v;
        scanf("%d%d",&u,&v);
        e[u].push_back(v);
        e[v].push_back(u);
    }
    memset(son,-1,sizeof(son));
    memset(width,0,sizeof(width));
    W=0;H=0;
    dfs(1,0);
    work(1,0,0,0);
    for(int i=1;i<=n;++i)
        printf("%d %d\n",px[i],py[i]);
    return 0;
}
```

Problem E. 击中目标

描述

在RPG游戏开发中，经常要判断子弹是否击中游戏目标。常用的方法是在3D场景中计算子弹射出的射线是否与目标物体相交。为了把问题简单化，我们假定子弹在距离目标物体$d(0 \leq d \leq 5000m)$的位置，以固定的初始速度$(v = 1000ms)$垂直射向目标物体所在平面。子弹仅受重力作用(重力加速度$g = 10m/s^2$)，忽略空气阻力。在子弹射出的同时，目标物体围绕物体中心点每秒钟以一定的角速度$a(0 \leq a \leq 180)$逆时针旋转，运动模型如下图所示。

子弹的初始位置用子弹在在目标物体所在平面的垂直投影坐标表示，而目标物体则用包围物



体的闭合折线所构成的多边形表示。我们规定，如果子弹射在目标物体的包围折线里面或在折线上，则表示击中，如果在折线外面则表示没有击中(精度要求$\epsilon \leq 10^{-6}$)。我们的任务是，给定目标物体的包围折线点，以及子弹的初始坐标、子弹到目标物体的距离、物体旋转角速度，要判断子弹是否击中目标。

注：目标物体的中心点可以用$x_c = (x_1 + x_2 + ... + x_m)/m$，$y_c = (y_1 + y_2 + ... + y_m)/m$计算得到。

输入要求

每组数据中第一行两个整数m，n，其中m表示包围折线坐标点的个数$(2 < m \leq 1000)$，n表示要判断的射击次数$(0 < n \leq 10^6)$。

接下来一行是m个坐标，按照$x_1y_1x_2y_2 \cdot x_my_m$方式排列，以空格分隔。所以，这一行共有$2m$个实数。

接下来是n行，每一行是四个实数，以空格分隔，分别是子弹初始坐标x，y，子弹到目标物体的距离d和目标物体的旋转速度a(以角度单位表示，$1^\circ = \pi/180^\circ \approx 0.0174532925$)。

输出数据

每组数据输出n行，如果子弹击中目标，输出YES，否则，输出NO，需要换行。

样例输入

```
4 3
0 0 10 0 10 10 0 10
3 4 0 0
5 15 1000 90
-2 2 500 180
```

样例输出

```
YES
YES
NO
```

Problem E. 题解

前言

首先，这次GMCPC，防AK题是F和T。出题时，为了避免太多作业题，过于单调，稍微增加一些难度，又要求不能再网上直接找到答案，所以在玩CF时(测试显卡性能而已)就想到把以前课程里的作业题改一下变成E题。起初设定为2-3星题，后面听闻各路高手来临，只好委托求援，有了F题和T题，同时把E题稍微改了一点，难度调高到3-4星，这是出题背景。第二，E题数据没有任何问题，但是下午很多人提交都是WA，有人怀疑数据有问题或者由于精度原因导致结果不唯一。但实际上了测试过了，这两方面问题都没有。不过为了避免不必要的麻烦，最终将所有数据都改成整数，距离为0。并且屏蔽了大部分边界测试数据，只保留了四组，其中前面3组数据是凑数的，纯粹为了让人开心一下的。

考虑到网络赛是可以上网的，所以预期很容易在网上找到类似方法，只是需要花点时间调试即可过。但不知道为什么，结果是这样。

下面题解中有一部分内容只是一些科普的背景知识，其实和题目本身没有任何关系，可绕路。

多边形(Polygon)

所谓多边形就是指一系列线段的集合，恰好形成一个圈，并且线段之间彼此互不交叉。通常会用一个点的集合表示，其中每个点是是一对(x, y)坐标。多边形的线段由点集中相邻的点连接，并且最后一个点和第一个点相连构成一个圈。例如，样例中给出的点集

$$(0,0) \quad (10,0) \quad (10,10) \quad (0,10)$$

恰好构成一个正方形。当然，并不是所有的点集列表都可以构成一个多边形，例如

$$(0,-10) \quad (0,10) \quad (10,0) \quad (-10,0)$$

构成的线段会彼此交叉。一般满足线段互不交叉的多边形称为简单多边形(simple polygon)，在上下文没有奇异的情况，默认就是简单多边形。(E题题面没有特意强调，是默认你们都会默认就是简单多边形。)

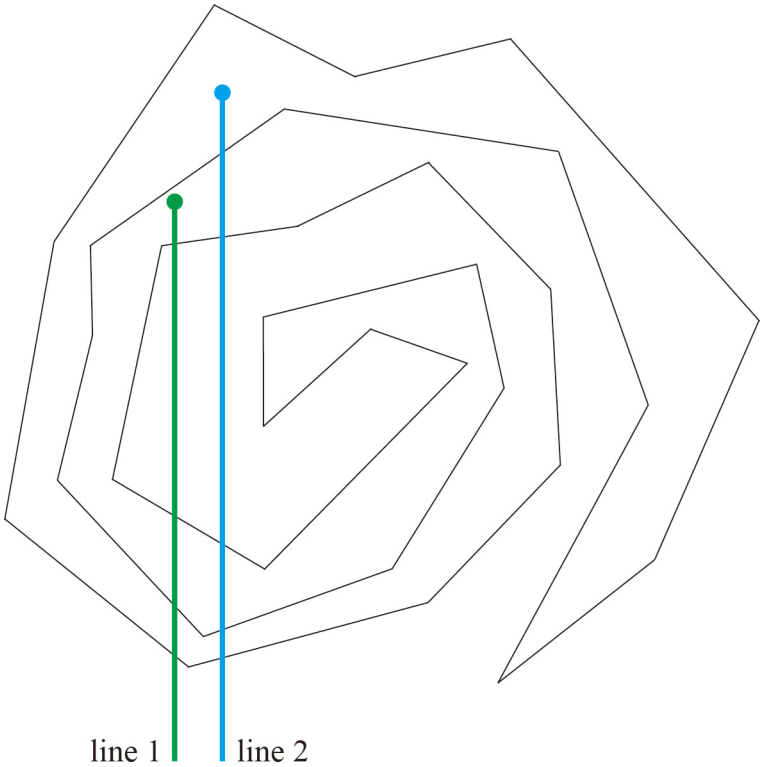
点包含问题(Point Inclusion in Polygon)

给定一个点，判断是在多边形内部还是外部。这个问题有很多方法可以做到，比如葛立恒扫描法(Graham's scan)等等。其中最简单最直观的可以用光线投射算法(Ray casting algorithm)。下面先介绍一下简单的背景。

根据约当曲线定理(Jordan curve theorem)，任何一个多边形都可以将平面切成两部分：内部区域和外部区域，并且任何从一个区域到另一个区域的道路都必然在某处与多边形的边相交。内部区域的面积总是有限，而外部区域包含无穷远的点。

若尔当曲线定理表面上看上去是显然的，但不是自明的，需要证明，而且证明它十分困难。对于较简单的闭曲线，例如多边形，是比较容易证明的，但要把它推广到所有种类的曲线，包括无处可微的曲线如科赫曲线，便十分困难。该定理对于球面上的若尔当曲线也成立，但对于环面上的若尔当曲线不成立。该定理最早由Camille Jordan (1838-1922)证明，但是有漏洞，最终由Oswald Veblen在1905年证明。

点包含问题非常有用。例如计算机渲染图形，需要判断内部点渲染着色，外部点不处理。对于离散有限个点的情形，图形学中通常采用种子填充算法。但连续情况，问题稍微复杂。对于一些足够简单点多边形，人的眼睛非常容易判断一个点是在内部还是外部，但是复杂一点，可能没那么容易(比如下图)。



采用光线投射算法比较容易直观解决这个问题。算法流程很简单，首先从这个点出发，向下(其实可以任何方向，有些人习惯向右水平方向，但我习惯向下，因为等会用x坐标计算交点比较方便)发射一条无穷远射线，统计射线与多边形的交点，如果交点数是偶数，则表示在外部，奇数则表示在多边形内部。例如下图中第一条直线line1，交点数是4，所以是外部。而第二条直线line2，交点数是5，所以在内部。

给定点$P_0(x_0, y_0)$，线段两个端点$P_1(x_1, y_1)$和$P_2(x_2, y_2)$。那么只需要判断$(x_1 < x_0 \&\& x_0 < x_2)$或者$(x_1 > x_0 \&\& x_0 > x_2)$即可。也可以写成看起来更简单的形式$(x_1 - x_0)(x_2 - x_0) < 0$。但在程序中最好不要这样判断，因为高精度的乘法会让程序很慢。在PC上可能问题不大，如果是嵌入式系统可能性能上有影响。

相交的情况有可能P_0恰好在线段上，所以需要计算一下交点$P_i(x_i, y_i)$，如果$y_0 = y_i$，则在

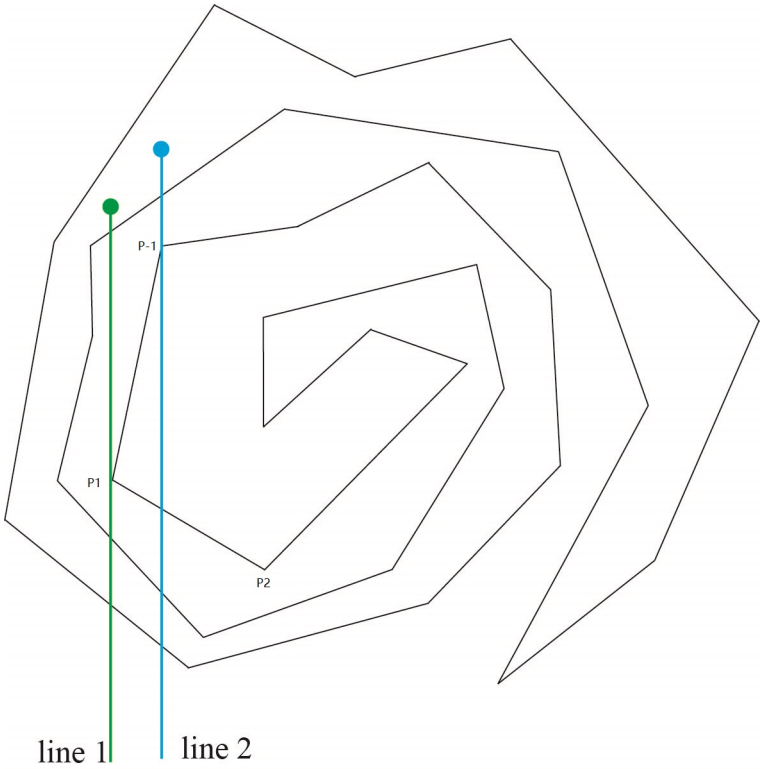
线段上。计算交点比较容易。首先写出线段方程

$$P = (1-t)P_1 + tP_2 \quad (0 \le t \le 1)$$

将x_0带入上述方程即可得到$t = (x - x_1)/(x_2 - x_1)$，再将$t$回带即可得到

$$y_i = \frac{(x - x_1)}{x_1 - x_2} y_1 + \frac{(x_1 - x_2)}{x_1 - x_2} y_2$$

情况比较复杂的是恰好通过端点时，如下图所示



直线`line1`与端点相交需要加2次，而直线`line2`与端点的交点只需要加1次。如何区别这两种情况？只需要一点小技巧，把直线想象稍微平移一点就可以区分开了。但实际上写程序时不需要真的去平移。稍微想象一下，P_1的前一个点P_{-1}，如果P_{-1}和P_2在直线同一侧，则加2，否则加1。是否同一侧只需判断$x_{-1} > x_0$即可。

实际我们只需要处理第一个端点P_1，因为在下一条线段中，P_2将变成第一个端点。此外，P_0上方的线段也是不用处理的，因为不可能相交。实际上，相交的线段极少，所以计算量很少，只需要简单的逻辑判断。拿张纸，画个图，想明白了。

关于旋转

题目中目标物体旋转，中心点只需要计算一次，因为中心点是不动点，不会旋转。平面绕点P_c的旋转可以想象为将点P_c移动坐标原点，旋转后，再平移到点P_c，可表示为

$$P' = \begin{pmatrix} \cos a & -\sin a \\ \sin a & \cos a \end{pmatrix} \left(P - P_c\right) + P_c$$

另外一点,不用真的去旋转多边形,而是去旋转子弹,两者是等价的,但时间复杂度是$O(1)$和$O(m)$的区别。

关于最小外接矩形

如果同一个多边形有多个点要判断，一个很自然的想法是预先计算多边形的最小外接矩形(minimum bounding box)，那样可以避免作很多无谓的计算。而这一步骤不会需要很多额外的开销，不管是时间还是空间。但实际上看了提交的代码，几乎很少人作这一步，可能提交的也少。

多边形有些特征也可以预先计算，从而优化计算时间，但是E题没有考虑这个问题。因为这不是重点，有兴趣的可以去查看一下资料。

后记

E题设置的时间是3s，因为标程是1.2s，不是故意卡时间。为了照顾Java和Python两位大哥，有几道题数据做了缩减。我想说的是，这种比赛最好用C/C++，因为出题人可能不懂Java。

我不懂ACM，E可能不符合你们胃口，抱歉。

附录

```
#include <iostream>
#include<limits>
#include<cmath>
using namespace std;
#define piover180 0.0174532925
#define epsilon 1e-6

inline bool eq(double  x, double y)
{
  return abs(x - y) < epsilon;
}
inline bool le(double x, double y)
{
```

```
  return (x < y) || (abs(x - y) < epsilon);
}
inline bool ge(double x, double y)
{
  return (x > y) || (abs(x - y) < epsilon);
}
class Point
{
 public:
  double x;
  double y;
  Point()
  {
    this->x = 0;
    this->y = 0;
  }
  Point(double x, double y)
  {
    this->x = x;
    this->y = y;
  }
};

class BoundingBox
{
 public:
  double xmin;
  double xmax;
  double ymin;
  double ymax;

  BoundingBox()
  {
    this->xmin = std::numeric_limits<double>::max();
    this->xmax = std::numeric_limits<double>::min();
    this->ymin = std::numeric_limits<double>::max();
    this->ymax = std::numeric_limits<double>::min();
  }
```

```
};

bool isInBoundingBox(BoundingBox& bbox, Point& point)
{
  return (ge(point.x, bbox.xmin) && le(point.x, bbox.xmax) && ge(
     point.y, bbox.ymin) && le(point.y, bbox.ymax));
}

inline void rotate(Point& p, Point& pc, double theta)
{
  double x, y;
  x = (p.x - pc.x) * cos(theta * piover180) - (p.y - pc.y) * sin(
     theta * piover180) + pc.x;
  y = (p.x - pc.x) * sin(theta * piover180) + (p.y - pc.y) * cos(
     theta * piover180) + pc.y;
  p.x = x;
  p.y = y;
}
bool isInPolygon(Point polygon[], int n, Point p)
{
  int crossings = 0;
  double t;
  double cy;
  int pre = 0;
  int current = 0;
  int next = 0;
  do {
    next = (current + 1) % n;
    pre = (current - 1) < 0 ? n - 1 : current - 1;
    if (((polygon[current].x < p.x) && (p.x < polygon[next].x)) ||
       ((polygon[current].x > p.x) && (p.x > polygon[next].x)))
    {
      t = (p.x - polygon[current].x) / (polygon[next].x - polygon[
         current].x);
      cy = (1-t) * polygon[current].y + t * polygon[next].y;
      if (eq(p.y, cy))
        return true;
      else if (p.y > cy)
```

```
        crossings++;
    }
    if (eq(polygon[current].x, p.x) && (le(polygon[current].y, p.y)
       ))
    {
      if (eq(polygon[current].y, p.y))
        return true;
      if (eq(polygon[next].x, p.x))
      {
        if (le(p.y, polygon[next].y))
          return true;
      }
      else if (polygon[next].x > p.x)
        crossings++;
      if (polygon[pre].x > p.x)
        crossings++;
    }
    current = next;
  } while (current != 0);

  return (crossings & 1);
}
int main()
{
  int m = 0;
  int n = 0;
  const double v = 1000;
  double t, d, theta;
  Point polygon[1000];
  BoundingBox bbox;
  Point p;
  Point pc;

  while (!cin.eof())
  //while (scanf("%d %d",&m,&n) != EOF)
  {
    cin >> m >> n;
    for (int i = 0; i < m; i++)
```

```
{
  cin >> p.x >> p.y;
  //scanf("%lf %lf",&p.x,&p.y);
  pc.x += p.x;
  pc.y += p.y;
  if (p.x < bbox.xmin)
  {
    bbox.xmin = p.x;
  }
  if (p.x > bbox.xmax)
  {
    bbox.xmax = p.x;
  }
  if (p.y < bbox.ymin)
  {
    bbox.ymin = p.y;
  }
  if (p.y > bbox.ymax)
  {
    bbox.ymax = p.y;
  }

  polygon[i] = p;
}
pc.x /= m;
pc.y /= m;
for (int i = 0; i < n; i++)
{
  cin >> p.x >> p.y >> d >> theta;
  //scanf("%lf%lf%lf%lf", &p.x, &p.y, &d, &theta);
  t = d / v;
  p.y -= 10 * t * t / 2;
  if (theta > 0)
    rotate(p, pc, theta * t);
  if(isInBoundingBox(bbox, p) && isInPolygon(polygon, m, p))
    cout << "YES\n";
  else
    cout << "NO\n";
```

```
    }
  }
  return 0;
}
```

Problem C. 今天星期几

关于闰年的问题，其实不用纠结，按照定义判断即可。现在使用格里历也是有误差。想要更精确的数据可以在NASA JPL找到星历表，目前比较新的有DE430，精度也仅仅覆盖1549-12-21到 2650-1-25。也有DE438，更新水星、火星和木星数据。宇宙的变化是混沌不可预料的，闰年也是我们自己的定义的，以为这样就能跟随太阳。

把日期转换为儒略历，避免去判断闰年。下列代码仅供参考(不是标程，题目不是我出的)。

```
#include<iostream>
using namespace std;
int main()
{
  int y, m, d, diff_day;
  while(!cin.eof())
  {
    cin >> y >> m >> d;
    m = (m + 9) % 12;
    y = y - m / 10;
    diff_day = 365 * y + y / 4 - y / 100 + y / 400 + (m * 306 + 5)
       / 10 + (d - 1) - 730425;
    cout << (diff_day / 4  + 1) <<" "<< (diff_day % 4 + 1) << endl;
  }
  return 0;
}
```

Problem N. 图像编码

这其实是四叉树编码和游程编码的混合。需要计算的是行列坐标转换为编码(其实就是Morton码)，这道题针对的是二维编码，只需要交叉一位二进制即可。有些高维编码可能都多个行列坐标，那样需要交叉多位。下面给出转换的示例代码，仅供参考(不是标程，题目不是我出的)。

```
unsigned int split_uint32_1(unsigned int x) {
  x &= 0x0000ffff; // x = ---- ---- ---- ---- fedc ba98 7654 3210
```

```
  x = (x ^ (x << 8)) & 0x00ff00ff; // x = ---- ---- fedc ba98 ---- ---- 7654 3210
  x = (x ^ (x << 4)) & 0x0f0f0f0f; // x = ---- fedc ---- ba98 ---- 7654 ---- 3210
  x = (x ^ (x << 2)) & 0x33333333; // x = --fe --dc --ba --98 --76 --54 --32 --10
  x = (x ^ (x << 1)) & 0x55555555; // x = -f-e -d-c -b-a -9-8 -7-6 -5-4 -3-2 -1-0
  return x;
}

unsigned long split_uint64_1(unsigned int x) {
  x &= 0x00000000ffffffffUL; // x = ---- ---- ---- ---- ---- ---- ---- ---- vuts rqpo nmlk jihg fedc ba98 7654 3210
  x = (x ^ (x << 16)) & 0x0000ffff0000ffffUL; // x = ---- ---- ---- ---- vuts rqpo nmlk jihg ---- ---- ---- ---- fedc ba98 7654 3210
  x = (x ^ (x << 8)) & 0x00ff00ff00ff00ffUL;  // x = ---- ---- vuts rqpo ---- ---- nmlk jihg ---- ---- fedc ba98 ---- ---- 7654 3210
  x = (x ^ (x << 4)) & 0x0f0f0f0f0f0f0f0fUL;  // x = ---- vuts ---- rqpo ---- nmlk ---- jihg ---- fedc ---- ba98 ---- 7654 ---- 3210
  x = (x ^ (x << 2)) & 0x3333333333333333UL;  // x = --vu --ts --rq --po --nm --lk --ji --hg --fe --dc --ba --98 --76 --54 --32 --10
  x = (x ^ (x << 1)) & 0x5555555555555555UL;  // x = -v-u -t-s -r-q -p-o -n-m -l-k -j-i -h-g -f-e -d-c -b-a -9-8 -7-6 -5-4 -3-2 -1-0
  return x;
}

inline unsigned int morton_code2(unsigned int x, unsigned int y)
{
  return  split_uint32_1(x << 1) | split_uint32_1(y);
}

inline unsigned int compact_uint32_1(unsigned int x) {
```

```
  x &= 0x55555555;  // x = -f-e -d-c -b-a -9-8 -7-6 -5-4 -3-2 -1-0
  x = (x ^ (x >> 1)) & 0x33333333;  // x = --fe --dc --ba --98 --76
      --54 --32 --10
  x = (x ^ (x >> 2)) & 0x0f0f0f0f;  // x = ---- fedc ---- ba98 ----
      7654 ---- 3210
  x = (x ^ (x >> 4)) & 0x00ff00ff;  // x = ---- ---- fedc ba98 ----
      ---- 7654 3210
  x = (x ^ (x >> 8)) & 0x0000ffff;  // x = ---- ---- ---- ---- fedc
      ba98 7654 3210
  return x;
}

inline unsigned int compact_uint64_1(unsigned long x) {
  x &= 0x5555555555555555UL;
  x = (x ^ (x >> 1)) & 0x3333333333333333UL;
  x = (x ^ (x >> 2)) & 0x0f0f0f0f0f0f0f0fUL;
  x = (x ^ (x >> 4)) & 0x00ff00ff00ff00ffUL;
  x = (x ^ (x >> 8)) & 0x0000ffff0000ffffUL;
  x = (x ^ (x >> 16)) & 0x00000000ffffffffUL;
  return x;
}

void morton_decode2(unsigned int c, unsigned int& x, unsigned int&
    y)
{
  x = compact_uint32_1(c >> 1);
  y = compact_uint32_1(c);
}
void morton_decode2(unsigned long c, unsigned long& x, unsigned
    long& y)
{
  x = compact_uint64_1(c >> 1);
  y = compact_uint64_1(c);
}
```

还可以用四叉树遍历欺骗这道题。

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

```
unsigned short current;
unsigned int count = 0;
//quadtree run length code
void quad_rlc(unsigned int** p, int n, int r, int c)
{
  if (n == 1)
  {
    if (current == p[r][c])
    {
      count++;
    }
    else
    {
      cout << b << "," << count << " ";
      current = p[r][c];
      count = 1;
    }
  }
  else
  {
    quad_rlc(p, n / 2, 0 + r, 0 + c);
    quad_rlc(p, n / 2, 0 + r, n / 2 + c);
    quad_rlc(p, n / 2, n / 2 + r, 0 + c);
    quad_rlc(p, n / 2, n / 2 + r, n / 2 + c);
  }
}
int main()
{
  int n = 0;
  int i, j;
  cin >> n;
  //new memory for large image
  unsigned int** p = new unsigned int* [n];
  for (i = 0; i < n; i++)
    p[i] = new unsigned int[n]();
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
```

```
    {
      cin >> p[i][j];
    }
  }
  current = p[0][0];
  count = 0;
  quad_rlc(p, n, 0, 0);
  cout << current << "," << count;

  for (i = 0; i < n; i++)
    delete[] p[i];
  delete[] p;
  return 0;
}
```

Problem J. 最小特征

这道题是我上课的C++作业题。很简单，图像旋转90度对于二进制码向右循环移动2位，图像翻转也就是异或运算对应于二进制反码。

```
#include <iostream>
#include<string>
inline unsigned char min(unsigned char a, unsigned char b){return a
    < b ? a : b;}
int main()
{
  unsigned char d;
  std::string s;
  int n = 0;
  while (!std::cin.eof())
  {
    std::cin >> n;
    for (int i = 0; i < n; i++)
    {
      std::cin >> s;
      d =   (s[0] - '0') * 128 + (s[1] - '0') * 64 + (s[2] - '0') *
          32
      + (s[5] - '0') * 16  + (s[8] - '0') * 8  + (s[7] - '0') * 4
```

```
      + (s[6] - '0') * 2    + (s[3] - '0');
      d = (s[4] == '0') ? d : (~d);
      std::cout  << (int)min(min(d, (d << 2) | (d >> 6)), min((d <<
          4) | (d >> 4), (d << 6) | (d >> 2))) << std::endl;
    }
  }
}
```

Problem G. 强迫症

重复元素大于一半，可以采用投票法，不需要排序。示例代码仅供参考(不是标程，题目不是我出的)。

```
#include <iostream>
using namespace std;
int main()
{
  int n = 0;
  int a;
  cin >> n;
  cin >> a;
  int repeat_element = a;
  int count = 1;
  for(int i = 1; i < n; i++)
  {
    cin >> a;
    if(count == 0)
    {
      repeat_element = a;
    }
    if(repeat_element == a)
      count++;
    else
      count--;
  }
  cout << repeat_element;
  return 0;
}
```