A

模拟比赛中的罚时判定，AC后每WA(没过都是WA)一次在现在时间基础上加20分钟，这题暴力模拟就好，但是没AC的题不计算罚时。

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int ac[20],wa[20];

int main(){

int T;

string t,s;

scanf("%d",&T);

while(T--){

memset(ac,0,sizeof(ac));

memset(wa,0,sizeof(wa));

int num,n,m;

scanf("%d%d",&n,&m);

int tim=0,cnt=0;

for(int i=0;i<m;i++){

cin>>num>>t>>s;

int ques=num-1000;

if(ac[ques]) continue;

if(s=="AC"){

ac[ques]=1;

cnt++;

tim+=(t[1]-'0')\*60+(t[3]-'0')\*10+(t[4]-'0');

tim+=wa[ques]\*20;

}

else wa[ques]++;

}

printf("%d %d\n",cnt,tim);

}

return 0;

}

B

除去一个数，使得数列的GCD(最大公因数)最大，首先需要知道数列的GCD如何计算，前n-1个数的GCD与第n个数取GCD可得这n个数的GCD。这题如果直接暴力，时间复杂度过大。可以把一个数前的GCD计算出来，再和这个数的后面的数取GCD，可得结果。先把1~n个数的GCD保存下来，分别从头尾保存，再遍历n次，取前i-1个数的GCD，与后n-i个数的GCD，可得结果。

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

const int N=100005;

int a[N],l[N],r[N];

int main(){

int t;

scanf("%d",&t);

while(t--){

memset(l,0,sizeof(l));

memset(r,0,sizeof(r));

int n;

scanf("%d",&n);

for(int i=1;i<=n;i++) scanf("%d",&a[i]);

l[1]=a[1];

r[n]=a[n];

for(int i=2;i<=n;i++) l[i]=\_\_gcd(l[i-1],a[i]);

for(int i=n-1;i>=1;i--) r[i]=\_\_gcd(r[i+1],a[i]);

int ans=max(l[n-1],r[2]);

//l[n-1]代表除去第n个数的GCD,r[2]代表除去第1个数的GCD

//l[i],r[i]分别代表前i个数的GCD与后i个数的GCD

for(int i=2; i<=n-1; i++) ans=max(ans,\_\_gcd(l[i-1],r[i+1]));

printf("%d\n",ans);

}

return 0;

}

C

*题意较难理解，重点是找一组边就好了，所以只需要判断无连接的点是否能连接。*

找**一组**边，使得**每个点**都有且只有一条边相连，每个点有两种状态：

1. 与前i个点各连接一条边 2.不连接，孤立

只需要每个2状态的点都能在后面找到一个1状态的点即可保证存在这样的一组边。若点的个数为奇数，每两个点连接一条边，一定存在一个点没有边连接，也不符合题意

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

int T;

scanf("%d",&T);

while(T--){

int n,cnt=1;

scanf("%d",&n);

for(int i=2;i<=n;i++){

int tmp;

scanf("%d",&tmp);

if(tmp==2) cnt++;

else cnt>0?cnt--:cnt++;

}

if(cnt!=0||n%2) printf("No\n");

else printf("Yes\n");

}

}

D

计算1^k+2^k+…+n^k再mod 1e9+7，分别取模加和即可，注意数据范围

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

int T;

scanf("%d",&T);

while(T--){

int n,m;

scanf("%d%d",&n,&m);

long long ans=0;

for(int i=1;i<=n;i++){

long long st=1;

for(int j=1;j<=m;j++){

st\*=i;

st%=(int)1e9+7;

}

ans+=st;

ans%=(int)1e9+7;

}

printf("%lld\n",ans);

}

}

E

奇数涂红，偶数涂绿，因为是连续的，所以红绿的区间数的差的绝对值不能超过1，注意这里红绿的格数都不能为0.

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

int r,g;

while(~scanf("%d%d",&r,&g)){

if(abs(r-g)>=2||r==0&&g==0) printf("NO\n");

else printf("YES\n");

}

return 0;

}

F

这里的特殊情况是情侣，所以先考虑情侣的数量，若为偶数则占N/2张桌子，其他人随意安排，（p+q+3）/4；若为奇数，则为（N-1/2）+1张桌子，安排一男一女与情侣坐，其余人为（p-1+q-1+3）/4；

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

int n,p,q,ans=1;

while(~scanf("%d%d%d",&n,&p,&q)){

if(n%2==0) ans=n/2+(p+q+3)/4;

else{

if(p>0) p--;

if(q>0) q--;

ans=(n-1)/2+(p+q+3)/4+1;

}

printf("%d\n",ans);

}

}

G

这里的要点是四舍五入，所以当面包价格累计到10\*k+6，7，8，9时，就买下面包，再计算9折后的价格

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

int n,k;

scanf("%d%d",&n,&k);

int sum=0,ans=0,tmp;

for(int i=1;i<=n;i++){

sum+=k;

if(sum%10>5){

ans+=sum\*9/10;

tmp=sum\*9%10;

if(tmp>=5) ans++;

sum=0;

}

}

ans+=sum\*9/10;

tmp=sum\*9%10;

if(tmp>=5) ans++;

printf("%d\n",ans);

}

H

N分解成3个质数和，如果暴力找3个数的话，会T，但是3个数其实并不一定需要互相牵制，可以先确定一个质数，再找后两个质数，时间复杂度就降下来了

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int is\_prime(int x){

if(x<=1) return 0;

for(int i=2;i<=sqrt(x);i++) if(x%i==0) return 0;

return 1;

}

int main(){

int n;

cin>>n;

if(n%2) printf("3 "),n-=3;

//奇数先确定一个质数3，选其他质数会有遗漏，只有2，3是两个相连的质数

else printf("2 "),n-=2;

//偶数先确定2

for(int i=2;;i++)

if(is\_prime(i)&&is\_prime(n-i)){

printf("%d %d\n",i,n-i);

return 0;

}

}