第八届ACM-ICPC暨CCCC-GPLT校内选拔赛初赛解题报告

Problem A: Problem Evaluation machine

题目大意：输入三个数 C,R,B，如果 $C=B$ 输出 `Accepted`，如果 $0<|C-B|\leq R$ 输出 `Wrong`，如果 $|C-B|>R$，输出`Extremely Wrong`。

题解：按照题目所写要求写程序即可， 注意多组输入的方式。

```
#include <bits/stdc++.h>
using namespace std;
int c, r;
int main()
{
    //freopen("1.in", "r", stdin);
    //freopen("1.out", "w", stdout);
    while(cin >> c >> r)
    {
        int t;
        scanf("%d", &t);
        if (t == c) puts("Accepted");
        else if ((int)abs(t-c) <= r) puts("Wrong");
        else puts("Extremely Wrong");
    }
    return 0;
}
```

Probelm B: CrazyX and His Money

题目大意: n 个账户, 每个账户有 a_i 的钱, a_i 可能不一样, 但是需要把他们变成一样的, 每次可以对一个 a_i 减去 k , 问最后能不能让所有的 a_i 都是一样的?

分析: 题目中没有说取钱干嘛, 所有有部分同学可能有一些误解, 其实取钱只是为了让不同账户中的钱是一样的; 不管怎么减, 每个账户可以减去的钱都一定是 k 的倍数. 所以, 如果有一个账户中的钱不能够减到所有账户中的最小值的话, 那么就不能把所有的账户中的钱都变成一样的;

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

const int maxn = 505;
int a[maxn];

int main(){
    int n, k;
    while(scanf("%d%d", &n, &k) == 2){
        int mina = 1000;
        for(int i=0;i<n;i++){
            scanf("%d", &a[i]);
            mina = min(mina, a[i]);
        }
        int flag = 0;
        for(int i=0;i<n;i++){
            if((a[i] - mina) % k != 0){
                flag = 1;
                break;
            }
        }
        printf("%s\n", flag?"NO":"YES");
    }
    return 0;
}
```

Problem C: YZL recites alphabets

题目大意：输入一个字母 x 和一个整数 n，按照字母表的顺序，输出从x下一个字母开始的n个字母，注意字母表是循环的Z的下一个又从A开始，所有的字母都是大写字母。

题解： 因为字母表中一共就26个字母，对字母输出的方式用偏移量的方式比较方便，例如想输出第3个字母(从0开始), 那么`putchar('A'+2)`即可，就可以输出字母C。按照题目要求只需要在上述输出方式再对26取模，就是当前要输出的字母的偏移量。

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    //freopen("1.in", "r", stdin);
    //freopen("1.out", "w", stdout);
    char ch;
    int n, c;
    while (cin >> ch >> n)
    {
        c = ch - 'A';
        for(int i = 1; i <= n; i++)
        {
            char p = 'A' + ((c+i) % 26);
            putchar(p);
        }
        putchar('\n');
    }
    return 0;
}
```

Problem D: LETTers

题目意思很简单，就是输出"LETT"这四个字母的字符画，考虑到这四个字母的形状，为了减少复杂的边界情况，出题的时候就保证了N为奇数并且不小于5，可能碰到的坑点在题面中已经加粗加红，

> What's more, there should be an empty line after each letter.

然后直接输出就好了……

注意每个字母的字符画后面输出一个空行即可。

```
#include <bits/stdc++.h>
using namespace std;
int n, m, T;

void out(char x, int cnt = 1, bool end = false) {
    for (int i = 0; i < cnt; i += 1) cout << x;
    if (end) cout << endl;
}

int main()
{
    cin >> T;
    while (T--) {
        cin >> n;
        for (int i = 0; i < n - 1; i += 1) {
            out('X');
            out('.', n - 1, true);
        }
        out('X', n, true);
        cout << endl;

        for (int i = 0; i < n; i += 1) {
            if (i == 0 || i == n / 2 || i == n - 1) {
                out('X', n, true);
            } else {
                out('X');
                out('.', n - 1, true);
            }
        }
        cout << endl;

        out('X', n, true);
        for (int i = 1; i < n; i += 1) {
            out('.', n / 2);
            out('X', 1);
            out('.', n / 2, true);
        }
        cout << endl;

        out('X', n, true);
        for (int i = 1; i < n; i += 1) {
            out('.', n / 2);
            out('X', 1);
            out('.', n / 2, true);
        }
        cout << endl;
    }
    return 0;
}
```

Problem E: Game

题意：两个人Nbyby和Ncjcj玩游戏，他们对一条长为N的链轮流进行染色，首先Nbyby进行染色，他可以任意选择一个没有被染色的点染成白色，然后是Ncjcj染色，他可以任意选择一个没有被染色的点，然后与这个点相连的点以及它自身都会被染成黑色。直到任意一个人不能染色的时候游戏终止，如果有某一个点为白色，那么Nbyby赢得游戏，否则Ncjcj赢得游戏。问如果两人都采取最优策略，谁将取得胜利。

先从简单的情况开始考虑：

$N=1$的时候，显然Nbyby获胜；

$N=2$的时候，由于Nbyby先手并且只能染一个点，那么Ncjcj后手随便染色一个点，由于“传染”的性质，会把两个点都染为黑色，Ncjcj获胜；

$N=3$的时候，如果编号为$1,2,3$，那么Nbyby只需要染色2号点，无论Ncjcj染1号或者3号点，都只能让两个点变成黑色，Nbyby接着只需要染色剩下的一个点即可获胜。

在这里你可以注意到一个性质，假设有一条链$1,2,3,4...$，如果Nbyby对2号点进行染色，那么Ncjcj就必须接着染1号点，因为如果Ncjcj这一步不染1号点，下一步只要Nbyby把1号点染成白色，那么Ncjcj就再也无法把1号点变成黑色，Ncjcj必输。

随着上述过程的进行，考虑$N\geq 5$且N为奇数的情况，每次按照上述过程链长度减2，直到长度为3时，按照之前的讨论，长度为3的时候Nbyby胜出；

接下来我们来讨论一下剩下的情况，即$N\geq 4$且N为偶数的情况：

$N=4$的时候Nbyby先手有没有必胜策略呢？答案是有的，对于$1,2,3,4$这条链，Nbyby只需要染色1号点或者4号点即可。那么为了使得这个白色点变黑，Ncjcj只能染色2号点或者3号点，但是这样会导致最后只剩下一个点，Nbyby接着只需要染色剩下的那个点即可获胜。

$N=6$的情况呢？同样Nbyby染色1号点，Ncjcj只能染色2号点，之后链长度变为3，于是规约到之前讨论的长度为3的情况，所以Nbyby获胜。

同理可得，$N\geq 4$且N为偶数情况下，都是Nbyby获胜。

综上所述，只需要$N=2$的时候输出Ncjcj，其余情况下输出Nbyby即可。

“玄不改命，氪不救非”。

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n, T;
    cin >> T;
    while (T--) {
        cin >> n;
        if (n != 2) cout << "Nbyby" << endl;
        else cout << "Ncjcj" << endl;
    }
    return 0;
}
```

Problem F: invoker

卡尔是dota的一个英雄，他可以通过合成三个能量球来创造出一个新技能，每个能量球都可以属于冰、火、雷三个属性之一，通过这种方法卡尔可以合成10个新技能出来。 现在问如果把属性数变成n,需要合成的球数变为m，那么卡尔可以合成多少新技能?

解法：这个题目的本质是从n个元素中有放回地取m次，求其组合数，公式为C_{n+m-1}^{n-1}

> 详 见 ： https://baike.baidu.com/item/%E9%87%8D%E5%A4%8D%E7%BB%84%E5%90%88/6774375?fr=aladdin

对于组合数的求解部分，因为数据范围比较小，直接暴力算阶乘即可。

```
#include <bits/stdc++.h>
using namespace std;
int n,m;
long long C(int a,int b)
{
    long long res=1;
    for(int i=0;i<b;i++)
        res=res*(a-i);
    for(int i=1;i<=b;i++)
        res/=i;
    return res;
}
int main()
{
    //freopen("../data/2.out","w",stdout);
    int T;
    cin>>T;
    for(int ca=1;ca<=T;ca++)
    {
        cin>>n>>m;
        cout<<C(n+m-1,n-1)<<endl;
    }
    return 0;
}
```

Problem G: winwine

题意: 给出一个 B 数组，告知B的产生方式$B[i] = A[i-1] + A[i] + A[i+1]$，现在让你反推$\sum_{i=1}^{n} A[i]$的值

解法: 对于该系数矩阵 A

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix}$$

当$n \bmod 3 = 2$时，该矩阵的秩为$n-1$，其他情况为n。 因为题目保证输入数据合法，所以总是可以通过高斯消元把原A数组解出来然后求A数组的和，但是本题还有一个取巧的方法。

我们只需要根据$n \bmod 3$的值，选出一个起点，当$n \bmod 3 = 0$时，起点从第二个位置开始，否则从第一个位置开始，然后每隔三个加起来，就能得到A数组的和了。

```
#include <bits/stdc++.h>
using namespace std;
int b[110];
int n;
int main()
{
    //freopen("../data/1.out","w",stdout);
    int T;
    cin>>T;
    for(int ca=1;ca<=T;ca++)
    {
        cin>>n;
        for(int i=1;i<=n;i++)
            cin>>b[i];
        long long ans=0;
        for(int i=(n%3?1:2);i<=n;i+=3)
            ans+=b[i];
        cout<<ans<<endl;
    }
    return 0;
}
```

Problem H: yby and buff

题目大意：yby玩了一个假期的LOL，这个游戏里的buff转移机制是这样的：

1. 英雄击杀野怪获得buff，持续60s;
2. 英雄死亡后身上的buff消失;
3. 击杀敌方英雄，可以夺取他的buff，并且持续时间刷新，即持续60s。

现在告诉你击杀关系，问所有英雄拥有buff的总时间是多少。

注意：

1. buff可以同时存在多个，如果0~10s内，x和y两个英雄都有buff，那么对答案的贡献是20s;
2. 英雄名字是'b'~'z'这25个小写字母，'a'是野怪。

做法：这是个麻烦点的模拟题。我们可以开一个 `last` 数组， `last[i]` 表示 `i` 号英雄(包括野怪)上次获得buff的时间， `last[i]<;0` 表示这个英雄没buff。然后按照时间顺序依次处理每条击杀关系，首先清算一下a_i和b_i在本次击杀前buff持续时间对答案的贡献，然后更新他们的 `last` 数组。所有击杀关系处理完后，再扫一遍 `last` 数组，把最后拿了buff没被杀的情况也算进答案中。

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

int n, last[30], ans;

void init() {
    ans = 0;
    v.clear();
    for (int i = 0; i < 30; i++)
        last[i] = -999;
}

int main() {
    //freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
    int cas; scanf("%d", &cas);
    while (cas --) {
        scanf("%d", &n);
        init();
        for (int i = 1; i <= n; i++) {
            int ti;
            char a[9], kill[9], b[9];
            scanf("%d%s%s%s", &ti, a, kill, b);
            last[0] = ti;
            if (last[b[0] - 'a'] < 0) continue;
            if (ti - last[b[0] - 'a'] <= 60) {
                ans += ti - last[b[0] - 'a'];
                if (last[a[0] - 'a'] >= 0)
                    ans += min(60, ti - last[a[0] - 'a']);
                last[a[0] - 'a'] = ti;
            } else ans += 60;
            last[b[0] - 'a'] = -999;
        }
        for (int i = 1; i < 26; i++)
            if (last[i] >= 0) ans += 60;
        printf("%d\n", ans);
    }

    return 0;
}
```

Problem I: Hungry

题目大意是有 n 个人，第 i 人 t_i 时刻到达一个饭店，t_i 互不相同。这个饭店一次只能做一顿饭。每个人，按照到达时间排成一个队列。 饭店每做完一顿饭之后马上处理队首的人的订单。 饭店 0 时刻开门，m 时刻关门，做一顿饭时长为 t，因此该饭店 $m-t$ 之后就不再接订单了。 这种情形下，CrazyX 可以选择任何一个时刻到，问需要等的最少的时间能够吃上饭。

分两种情况：

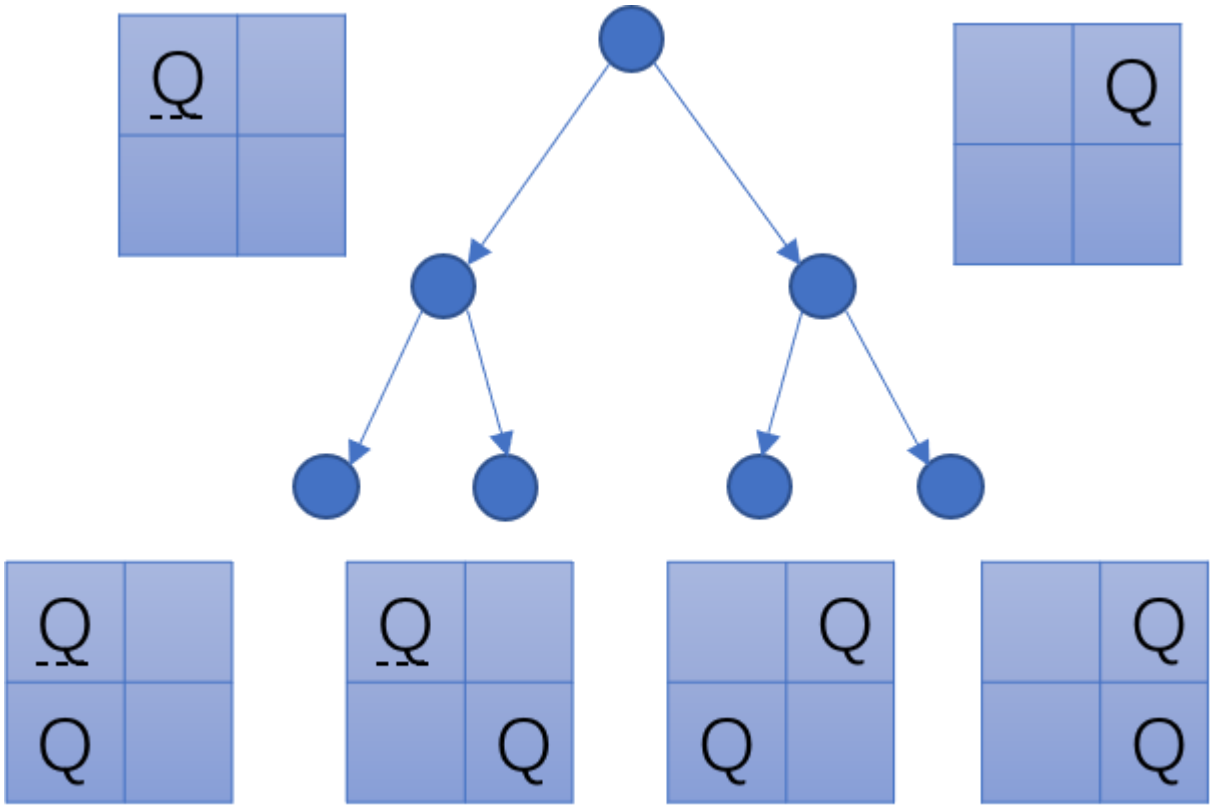
1. 饭店有空闲的时间
   这种情况，需要等的时间就是 P。需要注意的是，m 比较大的情况下，饭店做完 n 个人的饭后没有到 $m-t$，这时CrazyX到饭店饭店就可以马上为其做饭。
2. 饭店没有空闲时间
   在CrazyX不来的时候，我们可以得到每个人吃上饭的时间如$t, 2t, 3t, \cdots$。而CrazyX选择时间来的时候，CrazyX吃上饭的时间也是上述计算的时间中的某一个。这样，枚举CrazyX吃上饭的时间。对于每一个时间，我们可以算出他需要来的最晚的时间，做差更新答案就好。

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

using namespace std;
const int N = 100000;
const int INF = 0x3f3f3f3f;

int n, m, t;
int a[N], b[N];

int main(void) {
# ifdef owly
    //freopen("3.in", "r", stdin);
    //freopen("3.out", "w", stdout);
# endif

    int T;
    scanf("%d", &T);
    while (T --) {
        scanf("%d%d%d", &n, &m, &t);
        for (int i = 0; i < n; i++)
            scanf("%d", &a[i]);
        sort(a, a+n);
        for (int i = 0; i < n; i++)
            b[i] = a[i];

        bool f = false;
        int cur = 0;
        for (int i = 0; i < n; i++) {
            if (cur < b[i]) f = true;
            b[i] = max(cur, b[i]);
            cur = b[i] + t;
        }
        if (cur + t <= m) f = true;

        if (f) {
            printf("%d\n", t);
            continue;
        }

        int ans = INF;
        for (int i = 0; i < n; i++) {
            if (b[i] + t > m) break;
            ans = min(ans, b[i] - a[i] + 1 + t);
        }

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

    return 0;
}
```

Problem J: Horse Queen

题目大意是给一个 $n \times n$ 的棋盘，棋盘上可以放皇后，皇后可以横着走，斜着走，也可以向马一样走日格子，问对于一个 n 有多少种皇后的摆放方式，使得任何一个皇后都不能一步走到另外任何一个皇后的位置上。

因为 n 比较小，可以直接用回溯搜索的方式搜索。



当 n 等于 2 的时候，定义节点如上图所示。要求的就是最底层的节点中，满足上述条件的节点数有多少个。对于一个节点表示的皇后分布的图，可以用二维数组来表示。将数组的变化看成程序在这棵树上节点的遍历。我们按照如下的方法就可遍历整棵数的节点了。

```
int mp[N][N], n;                        // mp 表示当前节点代表的皇后分布，n 是格子的大小
void dfs(int k) {
  if (k == n) return;                   // k 是已经填完的层数，最开始用 dfs(0) 调用，为什么是
  else {                                // n 可以想一想
    for (int i = 0; i < n; i++) {       // 分别考虑皇后放在第 k 行的哪一个位置
      mp[k][i] = 1;                     // 放皇后
      dfs(k+1);                         // 转移状态
      mp[k][i] = 0;                     // 将皇后取下，为什么？大家可以参考上面的图想一想
    }
  }
}
```

这个题最暴力的方法是在 `if (k == n) return;` 的 `return` 的部分加上判断皇后分布是否合法的语句，这也不失为一种方法，但是肯定过不了这一个题。

这时就涉及到一个叫做**剪枝**的概念，其实很好理解。就是不用遍历到最底层的节点，而是某一个高层节点时判断冲突发生后就不用继续往下搜索了，举个例子就是放完第二行的皇后后，发现第一行和第二行的皇后是冲突的，那么这种情况下的第三行也就不用放了。更简单的，我可以在放皇后的时候就检查这个位置能不能放。比如看这一列有没有其他的皇后，对角线和反对角线上有没有其他的皇后等等，有就不放了，直接跳到下一个位置。

如何确定这个位置能不能放呢？可以遍历那个 `mp` 数组，这时可行的，但是用这个方法这个题还是会给超时。用 (i, j) 表示皇后的位置，显然在同一列，j 时一样的；也很容易发现，在同一对角线上，$i - j$ 是一样的；在同一反对角线上，$i + j$ 是一样的。因此，可以对于每一个列，对角线，反对角线设置一个变量，来保存是否存在一个皇后在这个位置上。当我们放皇后的时候和取下皇后的时候，这些数组才会变，更新一下就好了。

当然了，加上马的行走方式，直接检查 `mp` 数组就可以了。

```
# include <bits/stdc++.h>
# define cmax(x,y) x=max(x,y)
# define cmin(x,y) x=min(x,y)
using namespace std;
typedef double DB;
typedef long long LL;
typedef pair<int, int> Pr;

const int N = 15;
int ans[N];

int dx[] = {-1, -2, -2, -1};
int dy[] = {-2, -1, 1, 2};

int mp[N][N];
bool col[N], dm[2*N], ndm[2*N];
int cnt;
int n;
void dfs(int k) {
    if (k == n) cnt ++;
    else {
        for (int i = 0; i < n; i++) {
            if (col[i]) continue;
            if (ndm[i+k]) continue;
            if (dm[i-k+n]) continue;
            bool ok = true;
            for (int j = 0; j < 4; j++) {
                int nx = k + dx[j], ny = i + dy[j];
                if (0 <= nx && nx < n && 0 <= ny && ny < n && mp[nx][ny]) ok = false;;
            }
            if (!ok) continue;

            col[i] = 1;
            ndm[i+k] = 1;
            dm[i-k+n] = 1;
            mp[k][i] = 1;

            dfs(k+1);

            col[i] = 0;
            ndm[i+k] = 0;
            dm[i-k+n] = 0;
            mp[k][i] = 0;
        }
    }
}

int main(void) {
# ifdef owly
    freopen("in.txt", "r", stdin);
# endif
    memset(ans, -1, sizeof(ans));

    int T;
    scanf("%d", &T);
    while (T --) {
        scanf("%d", &n);
        if (ans[n] >= 0) printf("%d\n", ans[n]);
        else {
            cnt = 0;
            memset(col, 0, sizeof(col));
            memset(ndm, 0, sizeof(ndm));
            memset(dm, 0, sizeof(dm));
            dfs(0);
```

```
            ans[n] = cnt;
            printf("%d\n", ans[n]);
        }
    }

    return 0;
}
```