Problem A. 大学新章

本题直接输出就行了。

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

int main(){
    cout << "Code lights up my campus journey!";
    return 0;
}
```

Problem B. 时光的数字密语

这个题也是直接按要求计算输出就行了。

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

int main() {
    int age;
    cin >> age;
    long long ans = age * 3.156 * 1e7;
    cout << ans << "\n";
    return 0;
}
```

Problem C. 纸牌游戏

这个题我们多手动模拟几次就能做出来，假如两个数的值为 a, b 我们第一轮，减去一半 $a \to \frac{a}{2}$，第二轮 $b- = \frac{a}{2}$，现在两个数都是原先的一半相等的情况，然后现在第三轮我们只需要全部减去让其中一个变为 0 就行了，所以最终的值就是原先的一半，但是我们需要特别考虑一下奇数的情况，我们发现奇数是原先值的一半向上取整是最优解，大家可以尝试手动模拟一下。

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

int main() {
    long long n;
    cin >> n;
    cout << (n + 1) / 2 << endl;
    return 0;
}
```

Problem D. 素数

判断素数也算是大家学习编程的过程中一个比较典型的例题了，我们直接先特别判断是否为 1 或 2，然后判断 n 是不是偶数（因为除了 2 的偶数肯定不是素数）然后再从 3 开始遍历看每一个奇数能否整除 n，我们只用遍历到 $\sqrt{n}$ 就行了，因为如果前面没有数能够整除 n 那么后面肯定也没有了，因为整除后的结果乘以这个数肯定要等于 n （两个数是 n 的因子）那么这个结果肯定在后半段。

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

bool isPrime(int n) {
    if (n <= 1) {
        return false;
    }
    if (n == 2) {
        return true;
    }
    if (n % 2 == 0) {
        return false;
    }
    for (int i = 3; i <= sqrt(n); i += 2) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

int main() {
    int n;
    cin >> n;
    if (isPrime(n)) {
        cout << "Yes\n";
    } else {
        cout << "No\n";
    }
    return 0;
}
```

Problem E. 回文数

这个题我们直接从 1 开始遍历，然后判断是否为回文数就行了，具体过程如下：

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

bool isPalindrome(int num) {
    int t1 = num;  // 保存原数，用于后续对比
    int t2 = 0; // 存储反转后的数字
```

```
    // 反转数字（如121→12→1→0，反转过程：0*10+1=1 → 1*10+2=12 → 12*10+1=121）
    while (num > 0) {
        t2 = t2 * 10 + num % 10; // 取当前数的个位，拼接到反转数后
        num = num / 10; // 去除当前数的个位（整数除法）
    }

    return t2 == t1;
}

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        if (isPalindrome(i)) {
            cout << i << "\n";
        }
    }
    return 0;
}
```

Problem F. 沽酒兑欢

这个题我们也直接按照题目要求模拟就行了，先看手上的资金能不能买，能买就全部拿来买酒，然后再看能不能换，然后买的新酒或者换的新酒酒瓶喝瓶盖统计出来看能不能继续换，直到不能继续换为止，代码具体如下：

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

int main() {
    // 资金m、初始空瓶k、初始瓶盖g、单瓶酒价格p
    int m, k, g, p;
    cin >> m >> k >> g >> p;
    int tot = 0;  // 记录最多能喝到的总酒数
    // 第一步：用资金买酒（若资金不足或m<0，无法购买）
    int buy = 0;
    if (m >= p) {  // 资金足够买至少1瓶酒
        buy = m / p;  // 计算能买的酒数（整数除法，取商）
    }
    tot += buy;  // 累加购买的酒数
    k += buy;            // 买的酒喝完，新增空瓶
    g += buy;            // 买的酒喝完，新增瓶盖
    // 第二步：循环兑换酒（空瓶≥2或瓶盖≥4时可兑换，直至无法兑换）
    while (true) {
        // 计算当前可兑换的酒数：空瓶兑换数 + 瓶盖兑换数
        int t1 = k / 2;  // 2空瓶换1酒
        int t2 = g / 4;      // 4瓶盖换1酒
        int x = t1 + t2;
        if (x == 0) {  // 无酒可换，退出循环
            break;
        }
```

```
        tot += x;  // 累加兑换的酒数
        // 更新空瓶数：兑换后剩余空瓶（k%2） + 新换酒喝完的空瓶（x）
        k = (k % 2) + x;
        // 更新瓶盖数：兑换后剩余瓶盖（g%4） + 新换酒喝完的瓶盖（x）
        g = (g % 4) + x;
    }

    // 输出最终最多能喝的酒数
    cout << tot << "\n";

    return 0;
}
```

Problem G. L-R长条

对于这个题，我们发现如果有嵌套的 L - R 如：LLRLRRLR，我们优先选择最里面的 LR 再选外层的，那么内层的 LR 的分数就能多次计算，如果先选外层那么内层的分数就只能计算一次，然后我们再通过前缀和和双指针优化就可以了。

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

void solve() {
    long long n;
    cin >> n;
    vector<long long> a;
    a.push_back(0);
    for (int i = 0; i < n; ++i) {
        long long x;
        cin >> x;
        a.push_back(a.back() + x);
    }
    string s;
    cin >> s;
    long long ans = 0;
    long long l = 0;
    long long r = n - 1;
    while (r > l) {
        // 找到左侧第一个不是'R'的位置
        while (l < n && s[l] == 'R') {
            l++;
        }
        // 找到右侧第一个不是'L'的位置
        while (r >= 0 && s[r] == 'L') {
            r--;
        }
        // 若存在有效配对，累加前缀和差值
        if (l < r) {
            ans += a[r + 1] - a[l];
```

```
            l++;
            r--;
        }
    }
    cout << ans << "\n";
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

Problem H. 异数

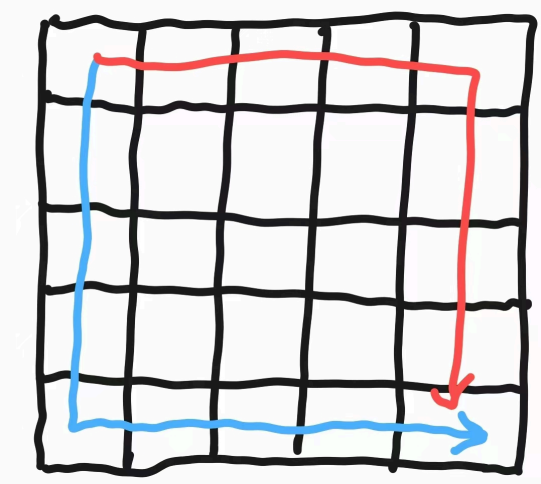
这个题就有点考经验和思维了，要找到一个数与数组里面的数不互为倍数关系，我们只需要找到一个大于数组里面所有数的一个素数（素数对于小于它的数只与 1 是倍数关系），然后只要数组里面没有 1 我们输出这个素数就行了，如果有 1 就输出 -1，然后我们观察数据范围，数组里面的数都是小于 10^9 的所以我们找一个大于 10^9 的素数就行了。

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

int main(){
    int n, t;
    cin >> n;
    bool ok = 1;
    long long x = 1000000007;
    for(int i = 0; i < n; i++){
        cin >> t;
        if(t == 1) ok = 0;
    }
    if(ok) cout << x << "\n";
    else cout << "-1\n";
    return 0;
}
```

Problem I. 走网格

对于这个问题我们观察到题目只有一个陷阱点，那么我们沿着边走，如图两种方案，一定有一种可行，在某一条路径上我们就输出另一条路径就行了。



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

int main() {
    int n, m;
    std::cin >> n >> m;
    int x = -1, y = -1;
    for (int i = 1; i <= n; ++i) {
        std::string s;
        std::cin >> s;
        for (int j = 1; j <= m; ++j) {
            if (s[j-1] == '#') {
                x = i;
                y = j;
            }
        }
    }

    bool p1 = (x == 1 || y == m);
    bool p2 = (y == 1 || x == n);

    if (!p1) {
        std::cout << std::string(m-1, 'D') + std::string(n-1, 'S') << "\n";
    } else {
        std::cout << std::string(n-1, 'S') + std::string(m-1, 'D') << "\n";
    }
    return 0;
}
```

Problem J. 游戏角色选择

这个我们只需要计算每个角色与主角的差异度，然后从小到大排序，然后选 m 个，输出最大的差异度就行了，也就是排序后的第 m 个差异度。

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

```cpp
using namespace std;

int main(){
    int n, m, k;
    cin >> n >> m >> k;
    vector<int> a(n);
    for (int i = 0; i < n; ++i) {
        cin >> a[i];
        a[i] = abs(a[i] - k);
    }
    sort(a.begin(), a.end());
    cout << a[m-1] << "\n";
    return 0;
}
```

Problem K. 点外卖

这个题按题目要求模拟判断输出就行了。

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

int main(){
    long long x;
    cin >> x;
    double pay8 = x*0.8;
    double pay5 = x - 5;
    if(pay8 < 0) pay8 = 0;
    if(pay5 < 0) pay5 = 0;
    if(pay8 < pay5){
        cout << "8";
    }
    else if(pay8 > pay5){
        cout << "5";
    }
    else {
        cout << "0";
    }
}
```

Problem L. 开灯

这个题我们从第一个灯开始各一个打开一个就行了，那么对于偶数个灯就刚好打开一半，奇数个就是一半上取整。

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

int main() {
    long long n;
    cin >> n;
    cout << (n + 1) / 2 << endl;
    return 0;
}
```

Problem M. 图案

拿一个数组，记录一下印章上其他的点和第一个点的相对位置。

然后我们从上往下扫那张纸，第一个出现的点一定是最顶上的点，然后进行遍历可以印到的点，判断是否是'x'，是的话将其改变，如果不是的话说明是不合法的，跳出就好啦。

```cpp
#include <algorithm>
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
int n,m,a,b;
char s[1006][1006],c;
int posx,posy,dx[1000006],dy[1000006],tot;
int main()
{
    scanf("%d%d%d%d",&n,&m,&a,&b);
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>s[i][j];
    for(int i=1;i<=a;i++)
    {
        for(int j=1;j<=b;j++)
        {
            cin>>c;
            if(!posx&&!posy&&c=='x')
            {
                posx=i,posy=j;
                continue;
            }
            if(c=='x')dx[++tot]=i-posx,dy[tot]=j-posy;
        }
    }
    bool flag=0;
    for(int i=1;i<=n;i++)
    {
        if(flag==1)break;
        for(int j=1;j<=m;j++)
        {
```

```
            if(flag==1)break;
            if(s[i][j]=='x')
            {
                for(int k=1;k<=tot;k++)
                {
                    if(s[i+dx[k]][j+dy[k]]=='x')s[i+dx[k]][j+dy[k]]='.';
                    else flag=1;
                }
            }
        }
    }
    if(flag)printf("NIE\n");
    else printf("TAK\n");
    tot=0;posx=0;posy=0;
    return 0;
}
```