Задача A. Двоичный поиск

В целом, нужно просто реализовать алгоритм с лекции. Удобнее всего для сравнения использовать предикат $x > a_i$.

```
n, k = [int(x) for x in input().split()]
a = [int(x) for x in input().split()]
b = [int(x) for x in input().split()]
for key in b:
    l = 0
    r = n
    while l + 1 < r:
        m = (r + l) // 2
        if a[m] > x:
            r = m
        else:
            l = m
    if a[l] == x:
        return('YES')
    else:
        return('NO')
```

Задача B. Рядом

Делаем бинпосик - элемент l всегда меньше искомого, элемент r всегда больше или равен. Стартовые значения -1 и $len(arr)$. Когда l и r сходятся после бинпоиска, проверяем оба индекса и находим оптимальный. Дополнительно аккуратно обрабатываем случай, если после бинпоиска $l = -1$ или $r = len(arr)$.

```
n, k = [int(s) for s in input().split()]
ar = [int(s) for s in input().split()]
queries = [int(s) for s in input().split()]

def b_search(elem):
    l, r = 0, len(ar)-1
    while l < r:
        m = (l + r) // 2
        if elem > ar[m]:
            l = m+1
        else:
            r = m
    if r == 0 or abs(ar[r] - elem) < abs(ar[r-1] - elem):
        return ar[r]
    return ar[r-1]


for q in queries:
    print(b_search(q))
```

Задача C. Отгадай число

Эта задача требует сделать бинпоиск с границами $l = 1$, $r = n + 1$, и если интерактор на запрос говорит, что число меньше, мы двигаем r, иначе l. В конце ответ лежит в l. В основном задача про аккуратность в отправке запросов и понимание того, как работает интерактивная задача.

```
import sys
```

```
n = int(input())
l = 0
r = n + 1

while r - l > 1:
    mid = (l + r) // 2
    print(mid)
    sys.stdout.flush()

    resp = input()

    if resp == '<':
        r = mid
    else:
        l = mid


print('! ', l)
```

Задача D. Квадратный корень и квадратный квадрат

Нам дана функция с квадратичной скоростью роста и являющаяся монотонной на $[0; \infty)$. Значит, для любого C ответ найдется на промежутке $[0; 10^5]$. По моногонной функции можно сделать бинпоиск с критерием $f(x) \geqslant C$. Чтобы не подбирать точность в бинпоиске, можно просто сделать 100 итераций и вывести, что получилось :)

```
from math import sqrt

C = float(input())
l = -1
r = sqrt(C)


def f(x):
    return x ** 2 + sqrt(x + 1) - C


while r - l > 1e-8:
    mid = (l + r) / 2
    if f(mid) < 0:
        l = mid
    else:
        r = mid

print(l)
```

Задача E. Корень кубического уравнения

Поскольку многочлен нечетной степени, то он стремится к бесконечности на бесконечности и к минус бесконечности на минус бесконечности. Поскольку он непрерывный, то между этими точками он точно пересекает 0! Так что достаточно делать бинпоиск и отличать положительную и отрицательную половину.

От задачи про квадратный корень есть два принципиальных отличия. Первое — сложнее найти границы бинпоиска. Второе — функция чередует отрицательную и положительную половину в зависимости от знака коэффициента a.

Чтобы решить вторую проблему, мы домножим все коэффициенты на -1, если $a < 0$. Чтобы решить вторую проблему, мы заметим что ax^3 «перевесит» остальные слагаемые, когда $ax^3 > 3bx^2$, $1 \cdot x^3 > 3 \cdot 1000 \cdot x^2$,откуда $|x| < 3000$. Так что можно было взять любые достаточно большие границы

```
a, b, c, d = [int(s) for s in input().split()]
b, c, d = b/a, c/a, d/a


def f(x):
    return x**3 + b*x**2 + c*x + d


l, r = -1e6, 1e6
while r - l > 1e-8:
    x = (l + r) / 2
    if f(x) < 0:
        l = x
    else:
        r = x

print(l)
```