1 Must to know in Midterm

1. Date/Time: 09:20 to 12:20, Nov. 01, 2024 (Fri.)
2. Location: Classroom (Main ED401)
3. Range: Chapter 1 to Chapter 8 (One-dimensional Array) of the textbook
4. The midterm exam will be conducted on the Online Judge platform: https://ntnueecpoj.ngrok.io
5. Please sign your name on the attendance sheet (provided by the TA).
6. During the exam, all printed materials may be consulted.
7. Access to any online resources and electronic files is prohibited during the exam.
8. Uploading malicious code to attack the server will result in a score of 0.
9. Even if you can't code, don't cheat.
10. You can cry, but don't cry too loudly.

2 Things you can do

1. Focus on the exam.
2. Request clarification on questions (the TA will only respond with "yes," "no," or "no comment").
3. Ask for translations of any questions you don't understand.

3 Grading

Each question will have a corresponding score that reflects the number of test cases associated with it. For example, a question worth 10 points will have 10 test cases. You will earn 1 point for passing any single test case. In total, there will be 100 test cases distributed across all questions, and each test case is scored independently. This means you don't need to write a perfect program to earn partial credit.

Problem 1. *(10pt) Determinant*
Calculation of the Determinant of a 3x3 Matrix Given a 3×3 *matrix A:*

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

The formula for calculating the determinant $|A|$ *is:*

$$|A| = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

Input Description
The program expects the user to input 9 elements ($0 < a_i < 10^3, a_i \in N$) corresponding to the entries of a 3×3 matrix.

Output Description
The program will output the determinant.

Sample Input	**Sample Output**
1 2 3 4 5 6 7 8 9	0

Table 1: Sample I/O

Reference Answer

```c
#include <stdio.h>

int main()
{
    int matrix[3][3];
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            scanf("%d", &matrix[i][j]);
        }
    }
    int det = matrix[0][0] * (matrix[1][1] * matrix[2][2] - matrix[1][2] * matrix[2][1]) -
              matrix[0][1] * (matrix[1][0] * matrix[2][2] - matrix[1][2] * matrix[2][0]) +
              matrix[0][2] * (matrix[1][0] * matrix[2][1] - matrix[1][1] * matrix[2][0]);
    printf("%d", det);
    return 0;
}
```

Problem 2. *(12pt) Jiro*
There are three brothers named A, B, and C. The age relationships among them are given by three characters S_{AB}, S_{AC}, S_{BC}, which mean the following:

- *If S_{AB} is $<$, then A is younger than B; if it is $>$, then A is older than B.*
- *If S_{AC} is $<$, then A is younger than C; if it is $>$, then A is older than C.*
- *If S_{BC} is $<$, then B is younger than C; if it is $>$, then B is older than C.*

Who is the middle brother, that is, the second oldest among the three?

Input Description
The input is given from Standard Input in the following format:
S_{AB} S_{AC} S_{BC}

Output Description
Print the name of the middle brother, that is, the second oldest among the three.

Sample Input	Sample Output
$<<<$	B

Table 2: Sample I/O

Reference Answer

```c
#include <stdio.h>

int main()
{
    char x, y, z;
    scanf(" %c %c %c", &x, &y, &z);
    if (x == '<' && y == '<' && z == '<')
    {
        printf("B\n");
    }
    else if (x == '<' && y == '<' && z == '>')
    {
        printf("C\n");
    }
    else if (x == '>' && y == '>' && z == '<')
    {
        printf("C\n");
    }
    else if (x == '>' && y == '>' && z == '>')
    {
        printf("B\n");
    }
    else if (x == '>' && y == '<' && z == '<')
    {
        printf("A\n");
    }
    else
    {
        printf("A\n");
    }
    return 0;
}
```

Problem 3. *(13pt) Three Small*
Given an array of N positive integers, find the third smallest number in the array. If there are duplicate numbers, they should be counted as different numbers. For example, in the array [5, 5, 5, 10], 5 is the smallest number and 5 is also the third smallest number.

Input Description
The first line contains an integer N ($3 \leq N \leq 10^3$), representing the size of the array. The second line contains N space-separated integers $A_i (1 \leq A_i \leq 10^7)$.

Output Description
Output a single integer representing the third smallest number in the array.

Sample Input	Sample Output
5 1 2 3 4 5	3

Table 3: Sample I/O

Reference Answer

```c
#include <limits.h>
#include <stdio.h>
#define MAXN 105
int arr[MAXN];
int findNextMin(int n, int min)
{
    int nextMin = INT_MAX;
    for (int i = 0; i < n; i++)
    {
        if (arr[i] > min && arr[i] < nextMin)
            nextMin = arr[i];
    }
    return nextMin;
}

int main()
{
    int N;
    scanf("%d", &N);
    for (int i = 0; i < N; i++)
        scanf("%d", &arr[i]);
    int firstMin = INT_MAX;
    for (int i = 0; i < N; i++)
    {
        if (arr[i] < firstMin)
            firstMin = arr[i];
    }
    int secondMin = findNextMin(N, firstMin);
    int thirdMin = findNextMin(N, secondMin);
    printf("%d\n", thirdMin);
    return 0;
}
```

Problem 4. *(15pt) Lucas Numbers*
Given a sequence of numbers, calculate the Lucas Numbers for each number in the sequence modulo 10000019. For each number x, find L(x) mod 10000019 where L(x) is the xth Lucas Numbers. Note that L(0) = 2, L(1) = 1.

$$L(n) = (L(n-1) + L(n-2))\%10000019$$

Input Description
The first line contains a single integer $n(1 < n < 10^7)$, representing the number of queries. The second line contains n space-separated integers $a_i(1 \leq a_i < 10^7)$, representing the position of Fibonacci numbers to calculate.

Output Description
Output n lines, where each line contains a single integer representing $L(a_i)$ mod 10000019 for the ith query.

Sample Input	**Sample Output**
3 1 2 3	2 1 3

Table 4: Sample I/O

Reference Answer

```c
#include <stdio.h>
#define MAXN 20000005
int lib[MAXN];
int main()
{
    lib[0] = 2;
    lib[1] = 1;
    for (int i = 2; i < MAXN - 5; i++)
        lib[i] = (lib[i - 1] + lib[i - 2]) % 10000019;
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        int x;
        scanf("%d", &x);
        printf("%d\n", lib[x]);
    }
    return 0;
}
```

Problem 5. *(15pt) Digit Counting*
Trung is bored with his mathematics homeworks. He takes a piece of chalk and starts writing a sequence of consecutive integers starting with 1 *to* N $(1 < N < 10000)$. *After that, he counts the number of times each digit (0 to 9) appears in the sequence. For example, with N = 13, the sequence is:*

12345678910111213

In this sequence, 0 appears once, 1 appears 6 times, 2 appears 2 times, 3 appears 3 times, and each digit from 4 to 9 appears once. After playing for a while, Trung gets bored again. He now wants to write a program to do this for him. Your task is to help him with writing this program.

Input Description
The input file consists of several data sets. The first line of the input file contains the number of data sets which is a positive integer and is not bigger than 20. The following lines describe the data sets. For each test case, there is one single line containing the number N.

Output Description
For each test case, write sequentially in one line the number of digit 0, 1, . . . 9 separated by a space.

Sample Input	Sample Output
2	0 1 1 1 0 0 0 0 0 0
3	1 6 2 2 1 1 1 1 1 1
13	

Table 5: Sample I/O

Hint : The last number output should not be followed by a space to avoid being judged as an incorrect answer.

Reference Answer

```c
#include <stdio.h>
int main()
{
    int tst, i, j, k;
    scanf("%d", &tst);
    while(tst--)
    {
        int n, ara[10];
        for(i=0; i<10; i++) ara[i]=0;
        scanf("%d", &n);
        for(i=1; i<=n; i++)
        {
            j = i;
            while(j!=0)
            {
                k=j%10;
                ara[k]++;
                j/=10;
            }
        }
        for(i=0; i<9; i++) printf("%d ", ara[i]);
        printf("%d\n", ara[9]);
    }
    return 0;
}
```

Problem 6. *(20pt) Standing On The Shoulders*
There are N giants, named 1 to N. When giant i stands on the ground, their shoulder height is A_i, and their head height is B_i.

You can choose a permutation $(P_1, P_2, \ldots, P_N)$ of $(1, 2, \ldots, N)$ and stack the N giants according to the following rules:

- *First, place giant P_1 on the ground. Giant P_1's shoulder will be at a height of A_{P_1} from the ground, and their head will be at a height of B_{P_1} from the ground.*
- *For $i = 1, 2, \ldots, N-1$ in order, place giant P_{i+1} on the shoulders of giant P_i. If giant P_i's shoulders are at a height of t from the ground, then giant P_{i+1}'s shoulders will be at a height of $t + A_{P_{i+1}}$ from the ground, and their head will be at a height of $t + B_{P_{i+1}}$ from the ground.*

Find the maximum possible height of the head of the topmost giant P_N from the ground.

Input Description

- The first line contains a single integer N ($2 \le N \le 2 \times 10^5$), representing the number of giants.
- The next N lines each contain two integers A_i and B_i ($1 \le A_i \le B_i \le 10^9$), where:
  - A_i represents the shoulder height of giant i when standing on the ground
  - B_i represents the head height of giant i when standing on the ground

Output Description
Output a single integer representing the maximum possible height.

Sample Input	Sample Output
3 4 10 5 8 2 9	18

Table 6: Sample I/O

If $(P_1, P_2, P_3) = (2, 1, 3)$, then measuring from the ground:

- giant 2 has a shoulder height of 5 and a head height of 8
- giant 1 has a shoulder height of 9 and a head height of 15
- giant 3 has a shoulder height of 11 and a head height of 18

The head height of the topmost giant from the ground cannot be greater than 18, so print 18.

Reference Answer

```c
#include <stdio.h>
#define MAXARRAYSIZE 2000005
int a[MAXARRAYSIZE], b[MAXARRAYSIZE], c[MAXARRAYSIZE];
int main()
{
    long long sum = 0;
    int i, n, max = 0;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d %d", &a[i], &b[i]);
        sum += (long long)a[i];
        c[i] = b[i] - a[i];
        if (max < c[i])
            max = c[i];
    }
    printf("%lld", sum + max);
    return 0;
}
```