1 Welcome to Computer Programming

In this assignment, I provide you with ample opportunities to write quality and substantial programs. The programs in the assignment are designed to be relatively complex, aimed at enhancing your programming skills. Unlike famous online judging systems such as zerojudge, UVA, LeetCode, and others, which often focus on algorithmic problems that require less code but demand significant techniques and problem-solving skills, the assignments here emphasize programming practice. As a result, the problems will have longer descriptions with comprehensive explanations on how to implement them. Your task is simply to realize these solutions using the C programming language. All problems will be tested with 5 sets of test data to ensure the correctness of your program, which may include data on the scale of millions.

In contrast, your midterm and final exams will differ in style from the assignments. They will involve more "design" elements, meaning you will need to think about "what kind of program to write to solve the problem." Typically, these types of problems can be solved with just a few lines of code and may have elegant solutions. However, they require substantial practice to develop the experience and intuition needed. Alongside your assignments, I recommend that you also work on the problem sets we provide, engage in diverse programming practice, and review the code written by your peers.

Before you start working on your assignment, here is a kind reminder. Please adhere to the following rules to avoid any potential penalties:

1. Avoid Plagiarism: While discussing ideas with classmates is encouraged, copying their code is not allowed.
   (If you use generative AI to help you with your assignments, it is obvious that you are very likely to end up with the same answers as others. In this case, we will directly judge it as plagiarism. Therefore, please "write" your assignments by yourself.)

2. Try Your Best but Not Too Much: You don't need to achieve a perfect score. Challenge yourself but understand that perfection isn't necessary.

Warning

1. Using arrays or any syntax not mentioned in class for this homework will result in a score of 0.

2. Please note that all submitted programs will be checked for plagiarism using the MOSS system. If your work is suspected of being copied, we will contact you for clarification. The final decision will be made by the instructor and teaching assistants, which could result in a zero score or reduced marks for the assignment.

2 Grading

Each problem will have a corresponding score. Some of the more challenging problems may have sub-problems, meaning you can earn partial points even if your solution isn't perfect. However, for most problems, you should assume that you need to pass all test cases to receive the full score for that problem.

3 Problems

Problem 1. *(10pt) Factorization*

For this question, it's a brief prompt: please write a program for prime factorization. To make the task more engaging and prevent students with a higher level of understanding from feeling bored, there will be 10 test cases, of which 5 are bonus questions. You need to think of some techniques to optimize your program.

Input Description

Let n be the input integer, where $1 \leq n < 10^{16}$.

Output Description

The output format is: $p_1^{a_1} * p_2^{a_2} * \ldots * p_k^{a_k}$ Where p_i are prime factors and a_i are the corresponding exponents. Factors are separated by asterisks (*).

Sample Input	Sample Output
20	2ˆ2 * 5
999997	757 * 1321

Table 1: Sample I/O

Hint

For Bonus, you can use the math header file if needed.

```c
#include <math.h>
```

Reference Answer

```c
#include <stdio.h>
#include <math.h>

int main() {
    int n, first = 1;
    scanf("%d", &n);
    int original_n = n;
    int count = 0;
    while (n % 2 == 0) {
        count++;
        n = n / 2;
    }
    if (count > 0) {
        if (count == 1) {
            printf("2");
        } else {
            printf("2^%d", count);
        }
        first = 0;
    }
    for (int i = 3; i <= sqrt(n); i = i + 2) {
        count = 0;
        while (n % i == 0) {
            count++;
            n = n / i;
        }
        if (count > 0) {
            if (!first) {
                printf(" * ");
            }
            if (count == 1) {
                printf("%d", i);
            } else {
                printf("%d^%d", i, count);
            }
            first = 0;
        }
    }
    if (n > 2) {
        if (!first) {
            printf(" * ");
        }
        printf("%d", n);
    }
    printf("\n");
    return 0;
}
```

Problem 2. *(10pt) Kirby ASCII ART*
In a world filled with creativity and technology, ASCII Art captivates countless artists and programmers with its unique charm. This art form cleverly utilizes basic ASCII characters, arranging and combining them to create beautiful images, allowing people to experience visual delight in the digital realm. As technology advances, this ancient art continues to evolve, shining brightly on various platforms.

One day, a young programming enthusiast decided to merge ASCII Art with the C language to create a brand-new application. He wanted to design a simple program that would read two integers representing the length and width, determining the shape of the ASCII Art. The user's input would become the canvas for his program, enabling him to depict unique character patterns within this matrix.

As his thoughts flowed, the young man devised a process that allowed users to input each character one by one, which would collectively form a complete ASCII Art piece. Each character held specific meaning; whether in shape or color, they were tangible expressions of the user's creativity.

However, the young man's program didn't stop there. He wanted users to see the numerical representations behind each character, so he added a feature that converted every entered character into its corresponding ASCII code, outputting these numbers line by line. This not only helped users understand the encoding of the characters but also added a layer of fun, making the creative process lively and engaging.

Input Description
First, two numbers W and H will be input (where $W \times H \leq 4 \times 10^4$), representing the width and height, respectively. Then, H lines will be input, each containing W characters C. Each character C is guaranteed to be one of the displayable ASCII characters.

Output Description
You need to convert all the input characters C into their ASCII code values. For example, 'A' will become 65. In total, there will be H lines, each containing W characters, which means you will output a total of $H \times W$ numbers.

Sample Input	**Sample Output**
W H Like Figure 1	W H Like Figure 2

Table 2: Sample I/O

Hint

1. You should first understand the input and output streams in the C language and how to handle buffers.
2. You should research and learn how to use the built-in C function 'getchar'.

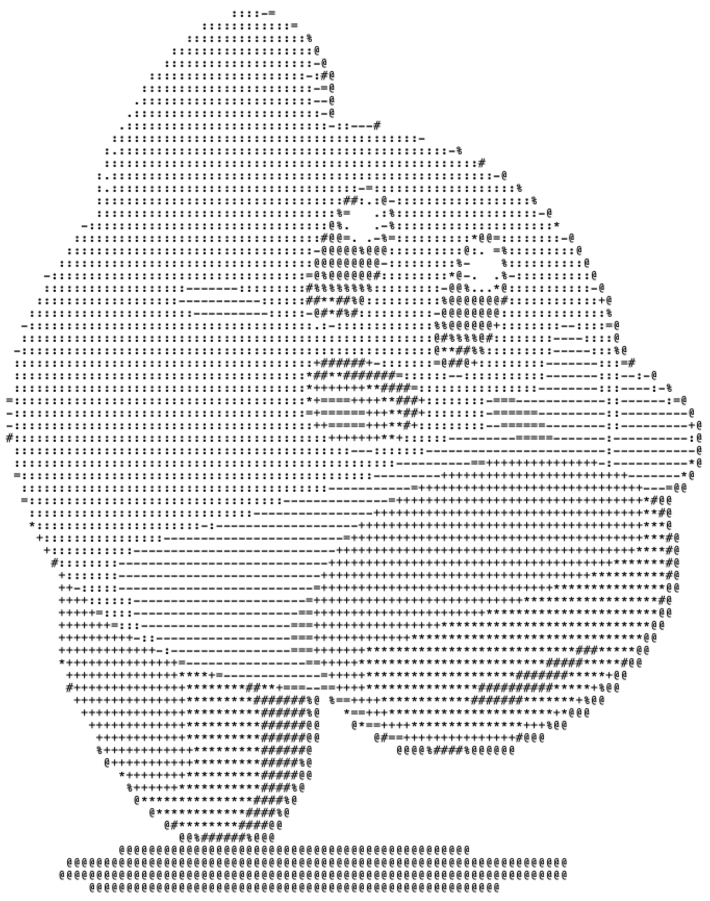

Figure 1: Kirby ASCII Art

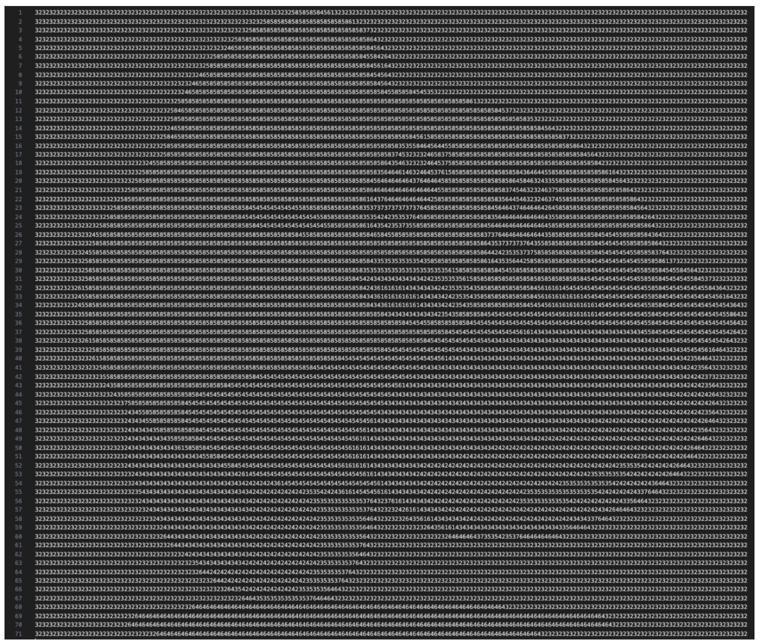

Figure 2: Your output will be

Reference Answer

```c
#include <stdio.h>
int main()
{
    int W, H;
    char C;
    scanf("%d %d", &H, &W);
    getchar();
    for (int i = 0; i < H; i++)
    {
        for (int j = 0; j < W; j++)
        {
            C = getchar();
            printf("%d", C);
        }
        printf("\n");
        getchar();
    }
    return 0;
}
```

Problem 3. *(10pt) Ditto*
Once upon a time in the vibrant world of Pokémon, a clever Pokémon Master set out on a quest unlike any other. This Master had a unique companion—Ditto, the Pokémon with the extraordinary ability to transform into any shape it desired. Today, however, was a special day. Ditto was ready to show off its skills, but there was a challenge involved.

The Master received an ASCII art representation of Ditto after its transformation, accompanied by a mysterious offset number. This offset held the key to unlocking the true form of Ditto. The Master understood that to reveal Ditto's complete form, they needed to adjust the ASCII code of each character in the art by adding this offset.

For instance, if the character 'A' (ASCII code 65) was part of the art and the offset was 2, the new character would be 'C' (ASCII code 67). Armed with this knowledge, the Pokémon Master took a deep breath and prepared to decode the art.

As the Master carefully processed each character, the transformed Ditto began to take shape. With every calculation, the excitement grew. Would Ditto become a powerful dragon? A shimmering fairy? Or perhaps a legendary creature from distant lands?

Finally, after intense focus and determination, the Master completed the task. The ASCII art blossomed into a magnificent new form, showcasing Ditto's versatility and the bond between Pokémon and Trainer.

The Master smiled, knowing that their cleverness had unlocked the true potential of Ditto, proving once again that with teamwork and ingenuity, anything is possible in the world of Pokémon. And so, they continued their adventures, ready to face whatever challenges awaited them next, with Ditto by their side.

Because the displayable ASCII codes are limited, we restrict your output characters to have ASCII codes between 32 and 127. In other words, you must take the modulus and add a certain number so that the ASCII code of your output characters falls within the range of 32 to 127.

Input Description
First, three numbers W, H and O will be input (where $W \times H \leq 4 \times 10^4$ and $O \leq 10^7$), representing the width, height and offset respectively. Then, H lines will be input, each containing W characters C. Each character C is guaranteed to be one of the displayable ASCII characters.

Output Description
You should output the transformed characters, which are the characters after adding the offset. In other words, you will output $W \times H$ characters to represent this ASCII art.

Sample Input	**Sample Output**
W H Like Figure 3	W H Like Figure 4

Table 3: Sample I/O

Hint

1. You should first understand the input and output streams in the C language and how to handle buffers.
2. You should research and learn how to use the built-in C function 'getchar'.

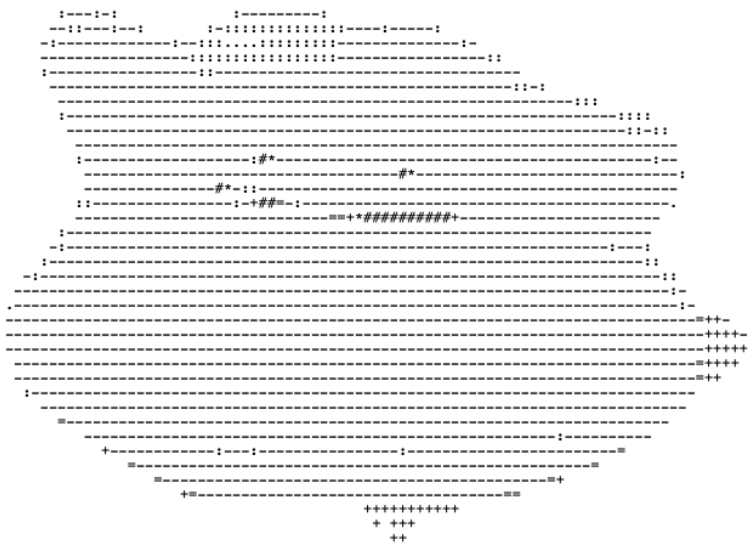

Figure 3: Kirby ASCII Art

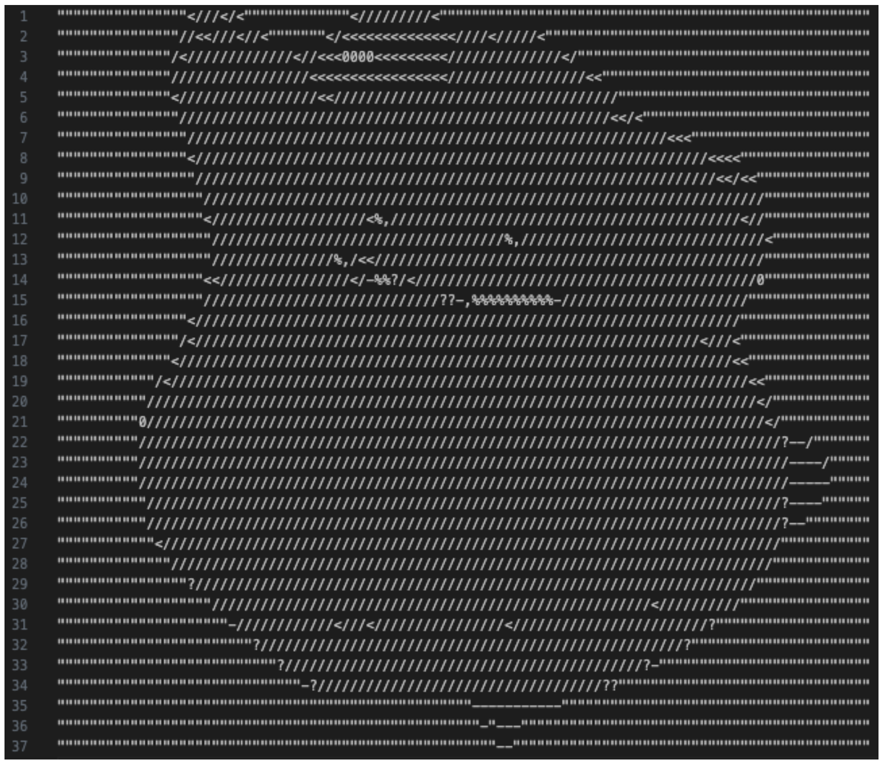

Figure 4: Your output will be

Reference Answer

```c
#include <stdio.h>
int main()
{
    int W, H, offset;
    char C;
    scanf("%d %d %d", &H, &W, &offset);
    getchar();
    for (int i = 0; i < H; i++)
    {
        for (int j = 0; j < W; j++)
        {
            C = getchar();
            int code = (int)(C + offset);
            if (code > 127)
                code = code % 127;
            if (code < 32)
                code += 32;
            printf("%c", code);
        }
        printf("\n");
        getchar();
    }
    return 0;
}
```

Problem 4. *(10pt) SAO World*

SAO (Solutions Always Optimal) is a game that started operations on November 6, 2022. In the game, there is an index called C-index. The C-index is a measure of a player's ability value (also known as the "starboom" value). The starboom value is calculated based on having N skills exceeding level N. Currently, there are 4 Beaters in the game, a term combining "beta-testers" and "cheaters." Newcomers believe that players who participated in the beta test can leverage the knowledge gained during the closed beta to gain an advantage in acquiring monster resources once the game officially launches.

Please write a C program to calculate the C-index for the 4 Beaters and sort them in descending order of their rankings.

Input Description

There will be a total of 4 rounds. In each round, an integer $N(0 \leq N \leq 10^7)$ will be provided, representing the number of skills of the i-th player. Following that, N numbers $S_i(0 \leq S_i \leq 10^9)$ will be given, representing the levels of the skills.

Output Description

Your task is to compute the C-index for each player and then output their character identifiers. The first player is represented by 'A', the second by 'B', and so on, with the last player represented by 'Z'.

If the sizes are the same, sort in descending lexicographical order.

Sample Input	Sample Output
1 1 1 2 1 3 1 4	D C B A
1 1 1 2 2 8763 8763 1 4	C D B A

Table 4: Sample I/O

Hint:

1. Although the problem mentions sorting, it is not necessarily required. You can iterate through all the players' C-indices and output the maximum each time to get the answer.

Figure 5: What SAO looks like

Reference Answer

```c
#include <stdio.h>

int main()
{
    int N, S;
    int A = 0, B = 0, C = 0, D = 0;
    int i;
    scanf("%d", &N);
    for (int i = 0; i < N; ++i)
    {
        scanf("%d", &S);
        if (S > A)
            ++A;
    }
    scanf("%d", &N);
    for (int i = 0; i < N; ++i)
    {
        scanf("%d", &S);
        if (S > B)
            ++B;
    }
    scanf("%d", &N);
    for (int i = 0; i < N; ++i)
    {
        scanf("%d", &S);
        if (S > C)
            ++C;
    }
    scanf("%d", &N);
    for (int i = 0; i < N; ++i)
    {
        scanf("%d", &S);
        if (S > D)
            ++D;
    }
    // Compare and print in descending order
    if (D >= C && D >= B && D >= A)
    {
        printf("D ");
        if (C >= B && C >= A)
        {
            printf("C ");
            if (B >= A)
            {
                printf("B A");
            }
            else
            {
                printf("A B");
            }
        }
        else if (B >= C && B >= A)
```

```c
            {
                printf("B ");
                if (C >= A)
                {
                    printf("C A");
                }
                else
                {
                    printf("A C");
                }
            }
            else
            {
                printf("A ");
                if (C >= B)
                {
                    printf("C B");
                }
                else
                {
                    printf("B C");
                }
            }
        }
        else if (C >= D && C >= B && C >= A)
        {
            printf("C ");
            if (D >= B && D >= A)
            {
                printf("D ");
                if (B >= A)
                {
                    printf("B A");
                }
                else
                {
                    printf("A B");
                }
            }
            else if (B >= D && B >= A)
            {
                printf("B ");
                if (D >= A)
                {
                    printf("D A");
                }
                else
                {
                    printf("A D");
                }
            }
            else
            {
                printf("A ");
                if (D >= B)
                {
                    printf("D B");
                }
                else
                {
                    printf("B D");
                }
            }
        }
        else if (B >= D && B >= C && B >= A)
        {
            printf("B ");
            if (D >= C && D >= A)
            {
                printf("D ");
                if (C >= A)
                {
                    printf("C A");
                }
                else
                {
                    printf("A C");
                }
            }
            else if (C >= D && C >= A)
            {
                printf("C ");
                if (D >= A)
                {
                    printf("D A");
                }
                else
                {
                    printf("A D");
                }
            }
            else
            {
                printf("A ");
                if (D >= C)
                {
                    printf("D C");
                }
                else
                {
                    printf("C D");
                }
            }
        }
        else
        {
            printf("A ");
            if (D >= C && D >= B)
            {
                printf("D ");
                if (C >= B)
                {
                    printf("C B");
                }
                else
                {
                    printf("B C");
                }
            }
            else if (C >= D && C >= B)
            {
                printf("C ");
                if (D >= B)
                {
                    printf("D B");
                }
                else
                {
                    printf("B D");
                }
            }
            else
            {
                printf("B ");
                if (D >= C)
                {
                    printf("D C");
                }
                else
                {
                    printf("C D");
                }
            }
        }
        return 0;
    }
```

Problem 5. *(10pt) GCD*

Given the value of N, you will have to find the value of G. The definition of G is given below :

$$G = \sum_{i=1}^{i<N} \sum_{j=i+1}^{j\leq N} GCD(i,j)$$

Here $GCD(i,j)$ means the greatest common divisor of integer i and integer j. For those who have trouble understanding summation notation, the meaning of G is given in the following code:

```
G=0;
for(i=1;i<N;i++)
  for(j=i+1;j<=N;j++)
  {
    G+=GCD(i,j);
  }
//GCD() is a function finds the greatest common divisor of the two numbers
```

Input Description

The input file contains at most 100 lines of inputs. Each line contains an integer $N(1 < N < 501)$. The meaning of N is given in the problem statement. Input is terminated by a line containing a single zero. This zero should not be processed.

Output Description

For each line of input produce one line of output. This line contains the value of G for corresponding N.

Sample Input	**Sample Output**
10 100 500 0	67 13015 442011

Table 5: Sample I/O

Figure 6: While you are doing your homework

Reference Answer

```c
#include <stdio.h>

int main()
{
    int N;
    while (1)
    {
        scanf("%d", &N);
        if (N == 0)
            break;
        long long G = 0;
        int i, j, a, b, temp;
        for (i = 1; i < N; i++)
        {
            for (j = i + 1; j <= N; j++)
            {
                a = i;
                b = j;
                while (b != 0)
                {
                    temp = b;
                    b = a % b;
                    a = temp;
                }
                G += a;
            }
        }
        printf("%lld\n", G);
    }
    return 0;
}
```