1 Must to know in Final Exam

1. Date/Time: 09:20 to 12:20, Dec. 20, 2024 (Fri.)
2. Location: Classroom (Main ED401)
3. Range: Chapter 8, 9, 11, 12 and 13 of the textbook
4. The 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 can be consulted.
7. Access to any on-line 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 cannot code, don't cheat.
10. You can cry, but do not 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 do not understand.

3 Grading

Each problem only if your submission achieves AC (Accepted), you can receive the full score; otherwise you will get 0 points. Your score will be compared with the following score chart:

Number of correct answers	0	1	2	3	4	5	6
Your final exam score	0	10	35	60	75	90	100

Table 1: Score Table

Problem 1. *Find the Median*

Given a one-dimensional array of integers, find and output the median value. The median is the middle value when the array is sorted in ascending order. If the array has an even number of elements, output the average of the two middle values rounded down to the nearest integer.

Input Description

The first line contains an integer N ($1 \leq N \leq 10^3$), representing the size of the array. The second line contains N space-separated integers, where each integer X satisfies ($-10^6 \leq X \leq 10^6$).

Output Description

Output a single integer representing the median value of the array.

Sample Input	Sample Output
5 1 7 3 4 2	3

Reference Answer

```c
#include <stdio.h>

void bubble_sort(int arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main()
{
    int n;
    scanf("%d", &n);

    int arr[1000];
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }

    bubble_sort(arr, n);

    if (n % 2 == 1)
    {
        printf("%d\n", arr[n / 2]);
    }
    else
    {
        printf("%d\n", (arr[n / 2 - 1] + arr[n / 2]) / 2);
    }

    return 0;
}
```

Problem 2. *Matrix Element Swap*
Given a matrix and a sequence of swap operations, determine the final state of the matrix after all swaps are performed. Each swap operation involves exchanging two elements specified by their row and column indices.

Input Description
The first line contains two integers N, M ($1 \leq N, M \leq 10^2$) representing the matrix dimensions. The next N lines each contain M integers, representing the initial matrix elements A_i ($0 \leq A_i \leq 10^6$). The next line contains an integer K ($1 \leq K \leq 10^3$) representing the number of swap operations. The next K lines each contain four integers r1, c1, r2, c2 ($0 \leq r1, r2 \leq N-1; 0 \leq c1, c2 \leq M-1$) representing the positions to swap.

Output Description
Print N lines, each containing M integers representing the final state of the matrix after performing all swap operations.

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

Reference Answer

```c
#include <stdio.h>

int main()
{
    int N, M;
    scanf("%d %d", &N, &M);

    int matrix[100][100];
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < M; j++)
        {
            scanf("%d", &matrix[i][j]);
        }
    }

    int K;
    scanf("%d", &K);

    for (int k = 0; k < K; k++)
    {
        int r1, c1, r2, c2;
        scanf("%d %d %d %d", &r1, &c1, &r2, &c2);

        int temp = matrix[r1][c1];
        matrix[r1][c1] = matrix[r2][c2];
        matrix[r2][c2] = temp;
    }

    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < M; j++)
        {
            printf("%d", matrix[i][j]);
            if (j < M - 1)
                printf(" ");
        }
        printf("\n");
    }

    return 0;
}
```

Problem 3. *Counting Inversions*

Your task is to write a program that counts the total number of inversions in a given sequence.

In a sequence of numbers, an inversion occurs when a pair of numbers is out of order, specifically, when a number that appears earlier in the sequence is greater than a number that appears later. For example, in the sequence [2, 4, 1], there are two inversions:

- (2,1): 2 appears before 1 but 2 bigger than 1
- (4,1): 4 appears before 1 but 4 bigger than 1

Input Description

The first line contains a single integer N ($1 \leq N \leq 10^3$), representing the length of the sequence. The second line contains N space-separated integers, where each integer (A_i $0 \leq A_i \leq 10^6$).

Output Description

Output a single integer representing the total number of inversions in the sequence.

Sample Input	Sample Output
5 2 4 1 3 5	3

Reference Answer

```c
#include <stdio.h>

int main()
{
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    long long inv_count = 0;
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            if (arr[i] > arr[j])
                inv_count++;
        }
    }
    printf("%lld\n", inv_count);
    return 0;
}
```

Problem 4. *String Pattern Matching*
Write a C program that finds all occurrences of a pattern string P in a text string T.

Input Description
The first line contains the text string T ($1 \leq |T| \leq 10^3$). The second line contains the pattern string P ($1 \leq |P| \leq |T|$). Both strings contain only lowercase English letters.

Output Description
Print all starting indices where pattern P appears in text T, in ascending order. If no match is found, print "NaN".

Sample Input	Sample Output
banana ana	1 3

Reference Answer

```c
#include <stdio.h>
#include <string.h>
char text[1005], pattern[1005];
int main()
{
    int found = 0;
    scanf("%s", text);
    scanf("%s", pattern);
    int text_len = strlen(text);
    int pattern_len = strlen(pattern);
    for (int i = 0; i <= text_len - pattern_len; i++)
    {
        int match = 1;
        for (int j = 0; j < pattern_len; j++)
        {
            if (text[i + j] != pattern[j])
            {
                match = 0;
                break;
            }
        }
        if (match)
        {
            if (found)
                printf(" ");
            printf("%d", i);
            found = 1;
        }
    }
    if (!found)
        printf("NaN");
    printf("\n");
    return 0;
}
```

Problem 5. *The Eagle Catches the Chickens*
In a peaceful farm, there are many baby chickens following Mother Hen in a line. But a hungry Eagle is watching them! You need to help Mother Hen keep track of her babies using a simple array-based system. When the Eagle catches a chicken, that chicken must leave the line. Mother Hen can also welcome new baby chickens to join the line at any position.

Input Description
The first line contains an integer N ($1 \leq N \leq 10^6$), representing the initial number of baby chickens. Their IDs are 1, 2, ..., up to N. For newly added chickens, their IDs will be $N+1, N+2, \ldots, M$ ($N \leq M \leq 10^8$). The following lines each contain one command:

- A: A baby chick joins at the end and is assigned the ID $N+k+1$, where k represents the number of newly added chicks.
- D x: The Eagle catches the baby chicken at position x
- P x y: Print the current line of chickens
- E: End of the day (game ends)

Guarantee that the number of commands P is less than 50.

Output Description
For each command 'P', print the current line of chickens: $x \to y \to z \to END$
x, y, z are the positions of chickens (starting from 1) If no chickens remain, print "Empty"
The program ends when 'E' command is received.

Sample Input	Sample Output
3	1 2 3
P	1 3
D 2	3
P	3
D 1	Empty
P	Empty
D 1	4 5
P	
D 3	
P	
P	
A	
A	
P	
E	

Note
In the example test, the ID of the first newly added chick is $N+k+1$, where $N=3$ (the input N) and $k=0$ (no chicks have been added yet). The ID of the second newly added chick is $N+k+1$, where $N=3$, $k=1$ (one chick was added previously) +1.

Reference Answer

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

#define MAX_SIZE 10001

int main()
{
    int *next = (int *)malloc(sizeof(int) * MAX_SIZE);
    int *prev = (int *)malloc(sizeof(int) * MAX_SIZE);
    int *val = (int *)malloc(sizeof(int) * MAX_SIZE);
    int *pos_to_node = (int *)malloc(sizeof(int) * MAX_SIZE);
    int *node_to_pos = (int *)malloc(sizeof(int) * MAX_SIZE);
    int free_head;
    int head = -1;
    int tail = -1;
    int count = 0;
    int next_id;
    char command;
    int pos, N;

    scanf("%d", &N);
    next_id = N + 1;

    if (N > 0)
    {
        head = 0;
        tail = N - 1;
        for (int i = 0; i < N; i++)
        {
            val[i] = i + 1;
            prev[i] = i - 1;
            next[i] = i + 1;
            pos_to_node[i + 1] = i;
            node_to_pos[i] = i + 1;
        }
        prev[0] = -1;
        next[N - 1] = -1;
        count = N;
    }

    for (int i = N; i < MAX_SIZE - 1; i++)
        next[i] = i + 1;
    next[MAX_SIZE - 1] = -1;
    free_head = N;

    while (1)
    {
        scanf(" %c", &command);
        if (command == 'E')
            break;

        switch (command)
```

```
        {
        case 'A':
            if (free_head != -1)
            {
                int new_node = free_head;
                free_head = next[free_head];
                val[new_node] = next_id++;
                prev[new_node] = tail;
                next[new_node] = -1;
                if (count == 0)
                    head = new_node;
                else
                    next[tail] = new_node;
                tail = new_node;
                count++;
                pos_to_node[count] = new_node;
                node_to_pos[new_node] = count;
            }
            break;

        case 'D':
            scanf("%d", &pos);
            if (count > 0 && pos <= count)
            {
                int curr = pos_to_node[pos];
                if (curr == head)
                {
                    head = next[curr];
                    if (head != -1)
                        prev[head] = -1;
                }
                else if (curr == tail)
                {
                    tail = prev[curr];
                    if (tail != -1)
                        next[tail] = -1;
                }
                else
                {
                    next[prev[curr]] = next[curr];
                    prev[next[curr]] = prev[curr];
                }
                if (pos < count)
                {
                    int last_node = pos_to_node[count];
                    pos_to_node[pos] = last_node;
                    node_to_pos[last_node] = pos;
                }
                next[curr] = free_head;
                free_head = curr;
                count--;
                if (count == 0)
                    head = tail = -1;
            }
```

```
            break;

        case 'P':
            if (count == 0)
                printf("Empty\n");
            else
            {
                int curr = head;
                while (curr != -1)
                {
                    printf("%d", val[curr]);
                    if (next[curr] != -1)
                        printf(" ");
                    curr = next[curr];
                }
                printf("\n");
            }
            break;
        }
    }

    free(next);
    free(prev);
    free(val);
    free(pos_to_node);
    free(node_to_pos);

    return 0;
}
```

Problem 6. *QA*

The programming course is about to end. We hope that regardless of whether you are good at programming or whether you have enjoyed it, you can provide us with some feedback to help improve the design of this course. Please answer the following questions.

Check List

Questions	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Course content	☐	☐	☐	☐	☐
Teaching pace	☐	☐	☐	☐	☐
Teaching method	☐	☐	☐	☐	☐
Homework difficulty	☐	☐	☐	☐	☐
Exam difficulty	☐	☐	☐	☐	☐
Supplementary materials	☐	☐	☐	☐	☐
TA's assistance	☐	☐	☐	☐	☐
I have learned something	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

Pros and Cons Please list three advantages and disadvantages (if any are missing, you may not receive the full points). You can answer in Chinese or English.

Your Comment Is there anything you'd like to say?