Date : 31.03.2025
Lakshana Baskaran
24011103026

LAB - 10
Priority Queue ADT (using heaps)

QUESTION 1:

A. Utilize C++ STL if needed. Write a C++ program to solve the following,

Polycarp was presented with some sequence of integers a of length n ($1 \le a_i \le n$). A sequence can make Polycarp happy only if it consists of **different** numbers (i.e. distinct numbers).

In order to make his sequence like this, Polycarp is going to make some (possibly zero) number of moves.

In one move, he can:

- remove the first (leftmost) element of the sequence.

For example, in one move, the sequence $[3, 1, 4, 3]$ will produce the sequence $[1, 4, 3]$, which consists of different numbers.

Determine the minimum number of moves he needs to make so that in the remaining sequence all elements are different. In other words, find the length of the smallest prefix of the given sequence a, after removing which all values in the sequence will be unique.

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of two lines.

The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the given sequence a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — elements of the given sequence a.

It is guaranteed that the sum of n values over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case print your answer on a separate line — the minimum number of elements that must be removed from the beginning of the sequence so that all remaining elements are different.

Date : 31.03.2025
Lakshana Baskaran
24011103026

Example

input Copy

```
5
4
3 1 4 3
5
1 1 1 1 1
1
1
6
6 5 4 3 2 1
7
1 2 1 7 1 2 1
```

output Copy

```
1
4
0
0
5
```

Note

The following are the sequences that will remain after the removal of prefixes:

- $[1, 4, 3]$;
- $[1]$;
- $[1]$;
- $[6, 5, 4, 3, 2, 1]$;
- $[2, 1]$.

It is easy to see that all the remaining sequences contain only distinct elements. In each test case, the shortest matching prefix was removed.

SOURCE CODE:

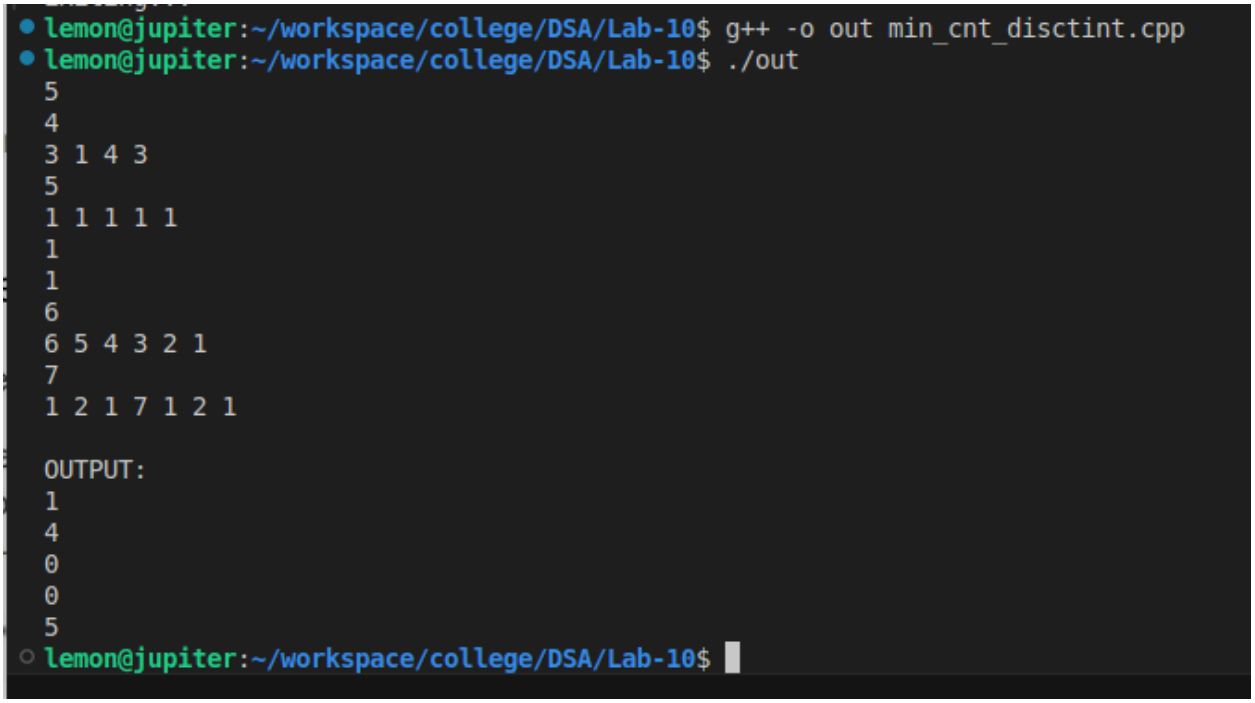
```
//to find the minimum number of times to delete the first element
from a sequence to get a sequence with unique numbers
#include <stdio.h>
#include <iostream>
#include <vector>
#include <stdbool.h>
using namespace std;

bool distinct(vector<int>);
int process(int);


int main() {
    int t, n, cnt;
    vector<int> out;

    cin >> t;
    for (int i = 0; i < t; ++i) {
        cin >> n;
```

```
        cnt = process(n);
        out.push_back(cnt);
    }

    cout << "\nOUTPUT:\n";
    for (int i = 0; i < out.size(); ++i) {
        cout << out[i] << endl;
    }

    return 0;
}

//checks if the sequence consists of distinct numbers or not
bool distinct(vector<int> vec) {
    for (int i = 0; i < vec.size(); ++i) {
        for (int j = 0; j < vec.size(); ++j) {
            if (i != j && vec[i] == vec[j]) {
                return false;
            }
        }
    }
    return true;
}

//counts the minimum number of deletions needed to make the sequence
distinct
int process(int n) {
    vector<int> vec;
    int x, count = 0;

    for (int i = 0; i < n; ++i) {
        cin >> x;
        vec.push_back(x);
    }

    while (vec.size() > 0) {
        if (distinct(vec) == true) {
            break;
        }
        vec.erase(vec.begin());
        count++;
    }
    return count;
```

Date : 31.03.2025
Lakshana Baskaran
24011103026

```
}
```

OUTPUT:



Date : 31.03.2025
Lakshana Baskaran
24011103026

QUESTION 2:

B. Utilize C++ STL if needed. Write a C++ program to solve the following,

Three guys play a game: first, each person writes down n distinct words of length 3. Then, they total up the number of points as follows:

- if a word was written by one person — that person gets 3 points,
- if a word was written by two people — each of the two gets 1 point,
- if a word was written by all — nobody gets any points.

In the end, how many points does each player have?

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 1000$) — the number of words written by each person.

The following three lines each contain n **distinct** strings — the words written by each person. Each string consists of 3 lowercase English characters.

Output

For each test case, output three space-separated integers — the number of points each of the three guys earned. You should output the answers in the same order as the input; the i-th integer should be the number of points earned by the i-th guy.

Example

input Copy

```
3
1
abc
def
abc
3
orz for qaq
qaq orz for
cod for ces
5
iat roc hem ica lly
bac ter iol ogi sts
bac roc lly iol iat
```

output Copy

```
1 3 1
2 2 6
9 11 5
```

Note

In the first test case:

- The word **abc** was written by the first and third guys — they each get 1 point.
- The word **def** was written by the second guy only — he gets 3 points.

SOURCE CODE:

```
//to find the minimum number of times to delete the first element
from a sequence to get a sequence with unique numbers
#include <stdio.h>
#include <iostream>
#include <vector>
```

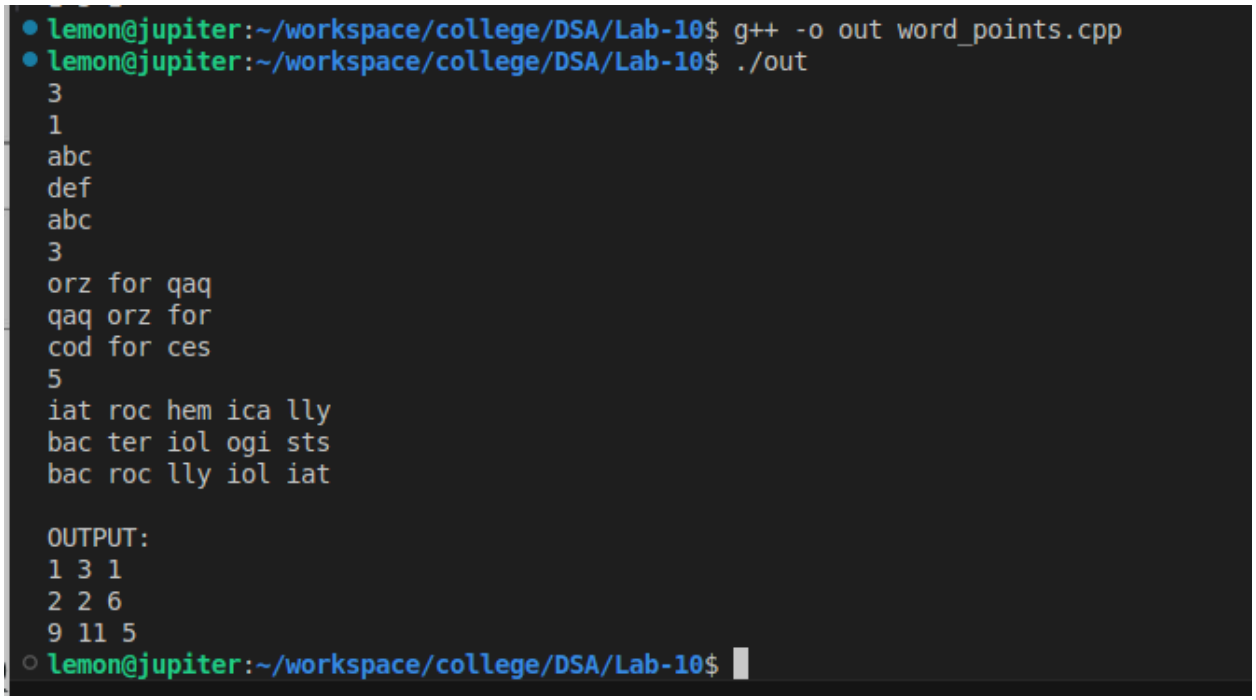
Date : 31.03.2025
Lakshana Baskaran
24011103026

```
#include <stdbool.h>
using namespace std;

vector<int> process(vector<string>, vector<string>, vector<string>);
bool in(string, vector<string>);
int points(vector<string>, vector<string>, vector<string>);

int main() {
    vector<string> joe;
    vector<string> bob;
    vector<string> ann;
    vector<string> temp;

    vector<int> points;
    vector<vector<int>> out;
    string x;
    int t, n;
    cin >> t;

    for (int k = 0; k < t; ++k) {
        joe = temp;
        bob = temp;
        ann = temp;
        cin >> n;
        for (int i = 0; i < n; ++i) {
            cin >> x;
            joe.push_back(x);
        }
        for (int i = 0; i < n; ++i) {
            cin >> x;
            bob.push_back(x);
        }
        for (int i = 0; i < n; ++i) {
            cin >> x;
            ann.push_back(x);
        }

        points = process(joe, bob, ann);

        out.push_back(points);
    }

    printf("\nOUTPUT:\n");
```

```
    for (int i = 0; i < t; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << out[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}


bool in(string str, vector<string> vec) {
    for (int i = 0; i < vec.size(); ++i) {
        if (str == vec[i]) {
            return true;
        }
    }
    return false;
}

int points(vector<string> vic, vector<string> ch1, vector<string>
ch2) {
    int pt = 0;
    for (int i = 0; i < vic.size(); ++i) {
        if (in(vic[i], ch1) == true && in(vic[i], ch2) == false) {
            pt = pt + 1;
        }

        if (in(vic[i], ch2) == true && in(vic[i], ch1) == false) {
            pt = pt + 1;
        }

        if (in(vic[i], ch1) == false && in(vic[i], ch2) == false) {
            pt = pt + 3;
        }
    }
    return pt;
}

vector<int> process(vector<string> joe, vector<string> bob,
vector<string> ann) {
    vector<int> pnts;

    pnts.push_back(points(joe, bob, ann));
```

```
    pnts.push_back(points(bob, joe, ann));
    pnts.push_back(points(ann, bob, joe));

    return pnts;

}
```

OUTPUT:



Date : 31.03.2025
Lakshana Baskaran
24011103026

QUESTION 3:

C. Write a separate C++ menu-driven program to implement Priority Queue ADT using a max heap. Maintain proper boundary conditions and follow good coding practices. The Priority Queue ADT has the following operations,

1. Insert
2. Delete
3. Display
4. Search
5. Sort (Heap Sort)
6. Exit

What is the time complexity of each of the operations? **(K4)**

SOURCE CODE:

```
//Implementation of priority Queue ADT using max heap
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <vector>
#include <stdbool.h>
using namespace std;

class PriorityQueue {
    private:
        vector<int> heap;

    public:
        void heapify(vector<int> &heap, int, int);
        void run_heap();
        void print();
        void insert(int);
        int deletion();
        bool Search(int);
        bool search(vector<int> &heap, int, int);
        void heapSort();

};
```

Date : 31.03.2025
Lakshana Baskaran
24011103026

```
int main() {
    PriorityQueue pq;
    int choice, value;
    vector<int> initial;
    cout << "\n--- Priority Queue Menu ---\n";
    cout << "1. Insert\n2. Delete Max\n3. Search\n4. Display\n5. Heap
Sort\n6. Exit\n";

    while (true) {
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to insert: ";
                cin >> value;
                pq.insert(value);
                break;
            case 2:
                cout << "Deleted max: " << pq.deletion() << endl;
                break;
            case 3:
                cout << "Enter value to search: ";
                cin >> value;
                if (!pq.Search(value))
                    cout << "Not found.\n";
                else
                    cout << "Found.\n";
                break;
            case 4:
                cout << "Heap contents: ";
                pq.print();
                break;
            case 5:
                pq.heapSort();
                cout << "Heap sorted: ";
                pq.print();
                break;
            case 6:
                printf("Exiting...\n");
                return 0;
            default:
                cout << "Invalid choice!\n";
```

```
        }
        cout << "\n\tThe queue: ";
        pq.print();
    }
}

void PriorityQueue::heapify(vector<int> &heap, int i, int n) {
    int largest = i;
    int left = 2*i + 1;
    int right = 2*i + 2;

    if (left < n && heap[left] > heap[largest]) {
        largest = left;
    }

    if (right < n && heap[right] > heap[largest]) {
        largest = right;
    }

    if (largest != i) {
        swap(heap[largest], heap[i]);
        heapify(heap, largest, n);
    }
}

void PriorityQueue::run_heap() {
    for (int i = heap.size()-1; i >= 0; --i) {
        heapify(heap, i, heap.size());
    }
}

void PriorityQueue::print() {
    for (int i = 0; i < heap.size(); ++i) {
        cout << heap[i] << " ";
    }
    cout << endl;
}


void PriorityQueue::insert(int key) {
    heap.push_back(key);
    run_heap();
}
```

Date : 31.03.2025
Lakshana Baskaran
24011103026

```
int PriorityQueue::deletion() {
    int dat = heap[0];
    heap[0] = heap[heap.size()-1];
    heap.erase(heap.begin()+heap.size()-1);
    run_heap();
    return dat;
}

bool PriorityQueue::Search(int key) {
    return search(heap, 0, key);
}

bool PriorityQueue::search(vector<int> &heap, int i, int key) {
    if (i >= heap.size()) {
        return false;
    }
    if (heap[i] == key) {
        return true;
    }
    bool leftSearch = search(heap, 2*i + 1, key);
    bool rightSearch = search(heap, 2*i + 2, key);

    return leftSearch || rightSearch;
}

void PriorityQueue::heapSort() {
    int n = heap.size();
    int key;
    vector<int> sorted;

    // One by one extract from heap
    for (int i = 0; i < n; ++i) {
        key = heap[0];
        heap.erase(heap.begin());
        run_heap();
        sorted.push_back(key);
    }
    heap = sorted;
}
```

Date : 31.03.2025
Lakshana Baskaran
24011103026

OUTPUT:

```
lemon@jupiter:~/workspace/college/DSA/Lab-10$ g++ -o out priority_q_adt.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-10$ ./out

--- Priority Queue Menu ---
1. Insert
2. Delete Max
3. Search
4. Display
5. Heap Sort
6. Exit

Enter your choice: 1
Enter value to insert: 2

        The queue: 2

Enter your choice: 1
Enter value to insert: 3

        The queue: 3 2

Enter your choice: 1
Enter value to insert: 6

        The queue: 6 2 3

Enter your choice: 1
Enter value to insert: 5

        The queue: 6 5 3 2

Enter your choice: 1
Enter value to insert: 9

        The queue: 9 6 3 2 5

Enter your choice: 2
Deleted max: 9

        The queue: 6 5 3 2

Enter your choice: 3
Enter value to search: 5
Found.

        The queue: 6 5 3 2

Enter your choice: 5
Heap sorted: 6 5 3 2

        The queue: 6 5 3 2

Enter your choice: 6
Exiting...
lemon@jupiter:~/workspace/college/DSA/Lab-10$
```

—