Practical No. 11 (Group E)

Name : Atharva B. Iparkar

Roll no : S211045

Class : S.E.

Div : A

Batch : A-2

Problem Statement :

Write program to implement a priority queue in C++ using an inorder list to store the

items in the queue. Create a class that includes the data items (which should be

template) and the priority (which should be int). The inorder list should contain these

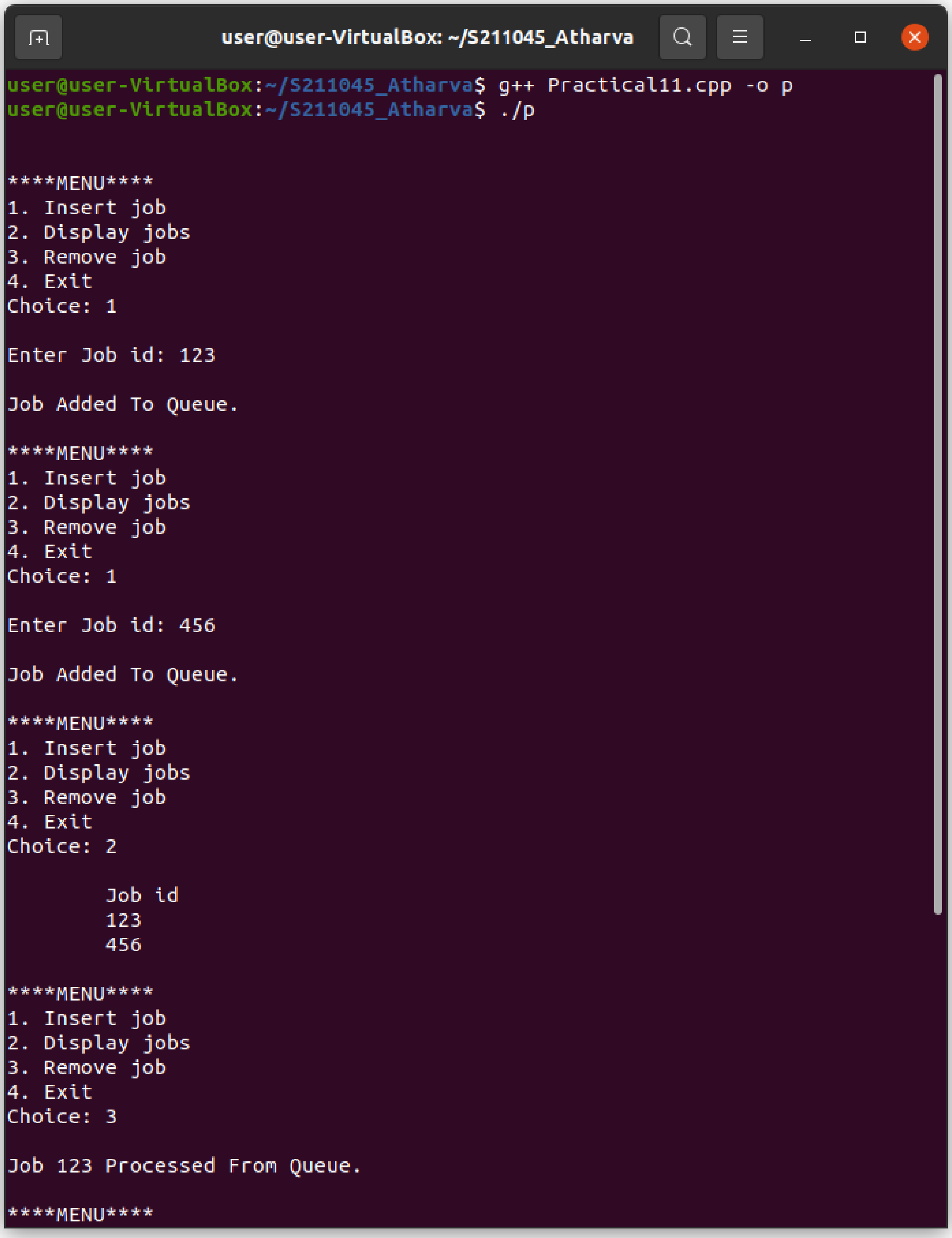
objects, with operator <= overloaded so that the items with highest priority appear at

the start of the list (which will make it relatively easy to retrieve the highest item.)

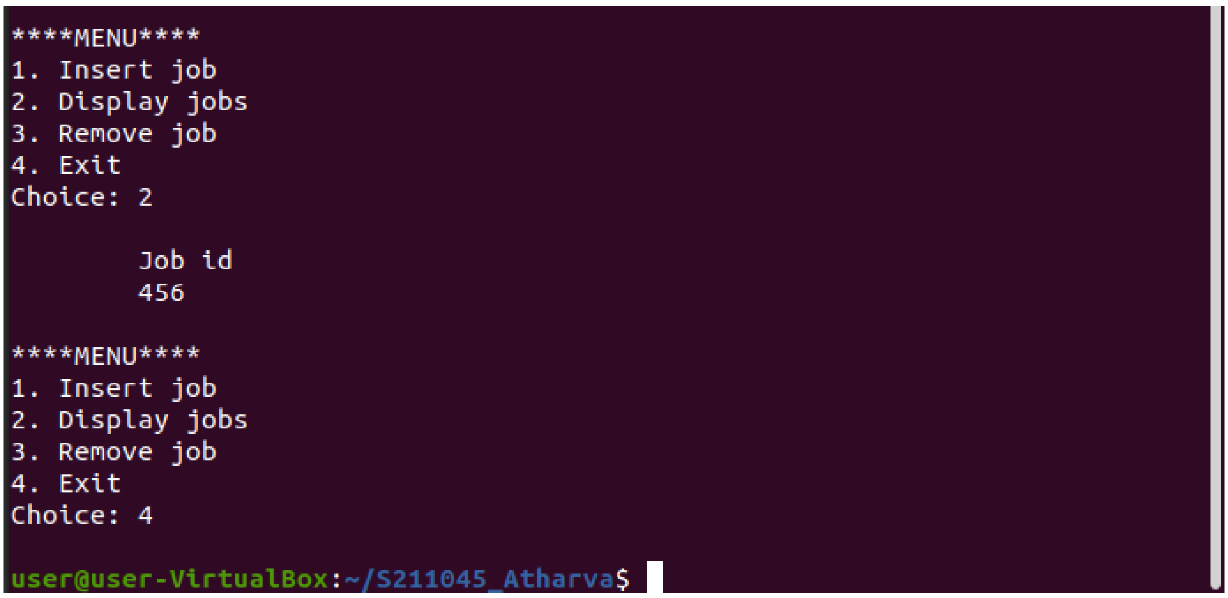
Code :

```
#include <iostream>

using namespace std;


const int MAX = 5;


class Job {
    int id;

    friend class Queue;

public:
    void getdata() {
        cout << "\nEnter Job id: ";
        cin >> id;
    }

    void putdata() {
        cout << "\n\t" << id;
    }
};

class Queue {
    int front, rear;
    Job queue[MAX];

public:
    Queue() {
        front = -1;
        rear = -1;
    }
    bool isEmpty();

    bool isFull();

    void insert();

    void remove();

    void display();

};


bool Queue::isEmpty() {

    return (front == -1 || front > rear);

}


bool Queue::isFull() {

    return (rear == MAX - 1);

}


void Queue::insert() {

    Job j;


    if (isFull()) {

        cout << "\nQueue is Full.";

    } else {

        j.getdata();

        if (front == -1) front = 0; // Set front to 0 if inserting the first job

        rear++;

        queue[rear] = j; // Insert job at the end
        cout << "\nJob Added To Queue.";
    }
}

void Queue::remove() {
    if (isEmpty()) {
        cout << "\nQueue is Empty.";
    } else {
        cout << "\nJob " << queue[front].id << " Processed From Queue.";
        front++;
        // Reset front and rear if the queue becomes empty after removal
        if (front > rear) {
            front = rear = -1;
        }
    }
}

void Queue::display() {
    if (isEmpty()) {
        cout << "\nQueue is Empty.";
    } else {
        cout << "\n\tJob id";
        for (int i = front; i <= rear; i++) {
            queue[i].putdata();
        }
    }
}

int main() {
    int ch;
    Queue q;

    do {
        cout << "\n\n****MENU****\n";
        cout << "1. Insert job\n";
        cout << "2. Display jobs\n";
        cout << "3. Remove job\n";
        cout << "4. Exit\n";

        cout << "Choice: ";
        cin >> ch;

        switch (ch) {
            case 1:
                q.insert();
                break;

            case 2:
                q.display();
                break;
            case 3:
                q.remove();
                break;

            case 4:
                cout << "\nExiting...";
                break;

            default:
                cout << "\nInvalid choice! Try again.";
        }
    } while (ch != 4);

    return 0;
}
```

Output :



```
user@user-VirtualBox:~/S211045_Atharva$ g++ Practical11.cpp -o p
user@user-VirtualBox:~/S211045_Atharva$ ./p


****MENU****
1. Insert job
2. Display jobs
3. Remove job
4. Exit
Choice: 1

Enter Job id: 123

Job Added To Queue.

****MENU****
1. Insert job
2. Display jobs
3. Remove job
4. Exit
Choice: 1

Enter Job id: 456

Job Added To Queue.

****MENU****
1. Insert job
2. Display jobs
3. Remove job
4. Exit
Choice: 2

        Job id
        123
        456

****MENU****
1. Insert job
2. Display jobs
3. Remove job
4. Exit
Choice: 3

Job 123 Processed From Queue.

****MENU****
```



```
****MENU****
1. Insert job
2. Display jobs
3. Remove job
4. Exit
Choice: 2

        Job id
        456

****MENU****
1. Insert job
2. Display jobs
3. Remove job
4. Exit
Choice: 4

user@user-VirtualBox:~/S211045_Atharva$ 
```