#include <iostream>

#include <deque>

#include <string>

using namespace std;

// Structure to represent a job with a priority

struct Job {

    int job\_id;

    string job\_name;

    int job\_priority;

    Job(int id, string name, int priority)

        : job\_id(id), job\_name(name), job\_priority(priority) {}

};

// Function to display job details

void displayJob(const Job& job) {

    cout << "Job ID: " << job.job\_id

         << ", Job Name: " << job.job\_name

         << ", Job Priority: " << job.job\_priority << endl;

}

class JobScheduler {

private:

    deque<Job> job\_queue; // Double-ended queue to store jobs

public:

    // Add a job to the deque

    void addJob(int id, string name, int priority, bool toFront = false) {

        Job new\_job(id, name, priority);

        if (toFront) {

            job\_queue.push\_front(new\_job); // Add high-priority job at the front

        } else {

            job\_queue.push\_back(new\_job); // Add job at the back

        }

        cout << "Job added successfully.\n";

    }

    // Remove a job from the deque (from front or back)

    void removeJob(bool fromFront = true) {

        if (job\_queue.empty()) {

            cout << "No jobs to remove.\n";

            return;

        }

        if (fromFront) {

            cout << "Removing job from front: ";

            displayJob(job\_queue.front());

            job\_queue.pop\_front();

        } else {

            cout << "Removing job from back: ";

            displayJob(job\_queue.back());

            job\_queue.pop\_back();

        }

    }

    // Display all jobs in the queue

    void displayJobs() {

        if (job\_queue.empty()) {

            cout << "No jobs available.\n";

            return;

        }

        cout << "Jobs in the queue:\n";

        for (const auto& job : job\_queue) {

            displayJob(job);

        }

    }

    // Search for a job by ID

    void searchJob(int id) {

        for (const auto& job : job\_queue) {

            if (job.job\_id == id) {

                cout << "Job found:\n";

                displayJob(job);

                return;

            }

        }

        cout << "Job with ID " << id << " not found.\n";

    }

};

int main() {

    JobScheduler scheduler;

    int choice, id, priority;

    string name;

    bool toFront;

    while (true) {

        cout << "\n--- Job Scheduling System ---\n";

        cout << "1. Add Job\n2. Remove Job\n3. Display Jobs\n4. Search Job\n5. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

        case 1:

            cout << "Enter Job ID: ";

            cin >> id;

            cout << "Enter Job Name: ";

            cin.ignore(); // To handle trailing newline character from previous input

            getline(cin, name);

            cout << "Enter Job Priority: ";

            cin >> priority;

            cout << "Add to front (1 for yes, 0 for no): ";

            cin >> toFront;

            scheduler.addJob(id, name, priority, toFront);

            break;

        case 2:

            cout << "Remove from front (1 for yes, 0 for no): ";

            cin >> toFront;

            scheduler.removeJob(toFront);

            break;

        case 3:

            scheduler.displayJobs();

            break;

        case 4:

            cout << "Enter Job ID to search: ";

            cin >> id;

            scheduler.searchJob(id);

            break;

        case 5:

            cout << "Exiting...\n";

            return 0;

        default:

            cout << "Invalid choice! Please try again.\n";

        }

    }

}