Name : Prem Sah
PRN : 123B1B234
Div : D

Assignment no. 10

Implement a job scheduling system for a manufacturing plant using a double-ended queue. The system needs to efficiently manage the processing of jobs on various machines throughout the plant. Each job has a Job_priority. The system should support the following operations:

a. Add Job
b. Remove Job
c. Display Job
d. Search Job

```
#include <iostream>
#include <string>
using namespace std;

class Job {
public:
    string name;
    int priority;
    Job* prev;
    Job* next;

    Job(string jobName, int jobPriority) {
        name = jobName;
        priority = jobPriority;
        prev = nullptr;
        next = nullptr;
    }
};

class Deque {
private:
    Job* front;
    Job* rear;

public:
    Deque() {
        front = nullptr;
        rear = nullptr;
    }

    void insertFront(string name, int priority) {
        Job* newJob = new Job(name, priority);
        if (front == nullptr) {
            front = rear = newJob;
        } else {
            newJob->next = front;
            front->prev = newJob;
            front = newJob;
        }
        cout << "Added job: " << name << " at the front." << endl;
    }

    void insertBack(string name, int priority) {
        Job* newJob = new Job(name, priority);
        if (rear == nullptr) {
            front = rear = newJob;
        } else {
            rear->next = newJob;
            newJob->prev = rear;
            rear = newJob;
        }
        cout << "Added job: " << name << " at the back." << endl;
    }

    void deleteFront() {
        if (front == nullptr) {
            cout << "No jobs to remove from the front." << endl;
            return;
        }
        Job* temp = front;
        cout << "Removed job: " << front->name << " from the front." << endl;
        front = front->next;
        if (front) {
            front->prev = nullptr;
        } else {
            rear = nullptr; // Queue is now empty
        }
        delete temp;
    }

    void deleteRear() {
        if (rear == nullptr) {
            cout << "No jobs to remove from the rear." << endl;
            return;
        }
        Job* temp = rear;
        cout << "Removed job: " << rear->name << " from the rear." << endl;
        rear = rear->prev;
        if (rear) {
            rear->next = nullptr;
        } else {
            front = nullptr; // Queue is now empty
        }
        delete temp;
    }

    void displayJobs() const {
        if (front == nullptr) {
            cout << "No jobs in the queue." << endl;
            return;
        }
        cout << "Jobs in the queue:\n";
        Job* current = front;
        while (current) {
            cout << "Job: " << current->name << ", Priority: " << current->priority << endl;
```

```
            current = current->next;
        }
    }
};

int main() {
    Deque jobQueue;
    int choice;
    string jobName;
    int jobPriority;

    do {
        cout << "\nJob Scheduling Menu:\n";
        cout << "1. Insert Job at Front\n";
        cout << "2. Insert Job at Back\n";
        cout << "3. Delete Job from Front\n";
        cout << "4. Delete Job from Rear\n";
        cout << "5. Display Jobs\n";
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter job name: ";
                cin >> jobName;
                cout << "Enter job priority: ";
                cin >> jobPriority;
                jobQueue.insertFront(jobName, jobPriority);
                break;
            case 2:
                cout << "Enter job name: ";
                cin >> jobName;
                cout << "Enter job priority: ";
                cin >> jobPriority;
                jobQueue.insertBack(jobName, jobPriority);
                break;
            case 3:
                jobQueue.deleteFront();
                break;
            case 4:
                jobQueue.deleteRear();
                break;
            case 5:
                jobQueue.displayJobs();
                break;
            case 6:
                cout << "Exiting the system." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 6);

    return 0;
}
```

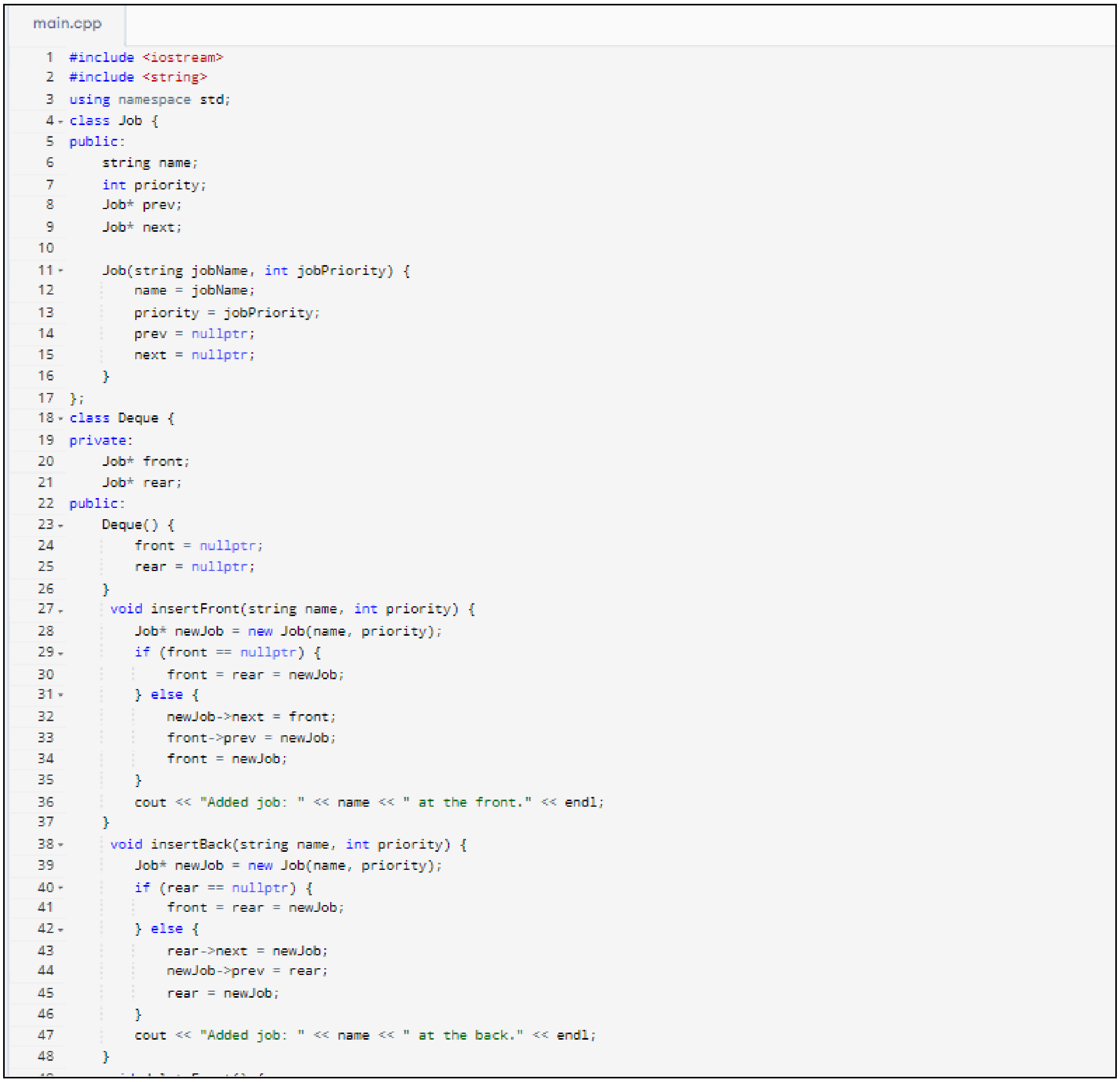
Output :

```
/tmp/MnQqqE3t9S.o

Job Scheduling Menu:
1. Insert Job at Front
2. Insert Job at Back
3. Delete Job from Front
4. Delete Job from Rear
5. Display Jobs
6. Exit
Enter your choice: 1
Enter job name: ABC
Enter job priority: 1
Added job: ABC at the front.

Job Scheduling Menu:
1. Insert Job at Front
```

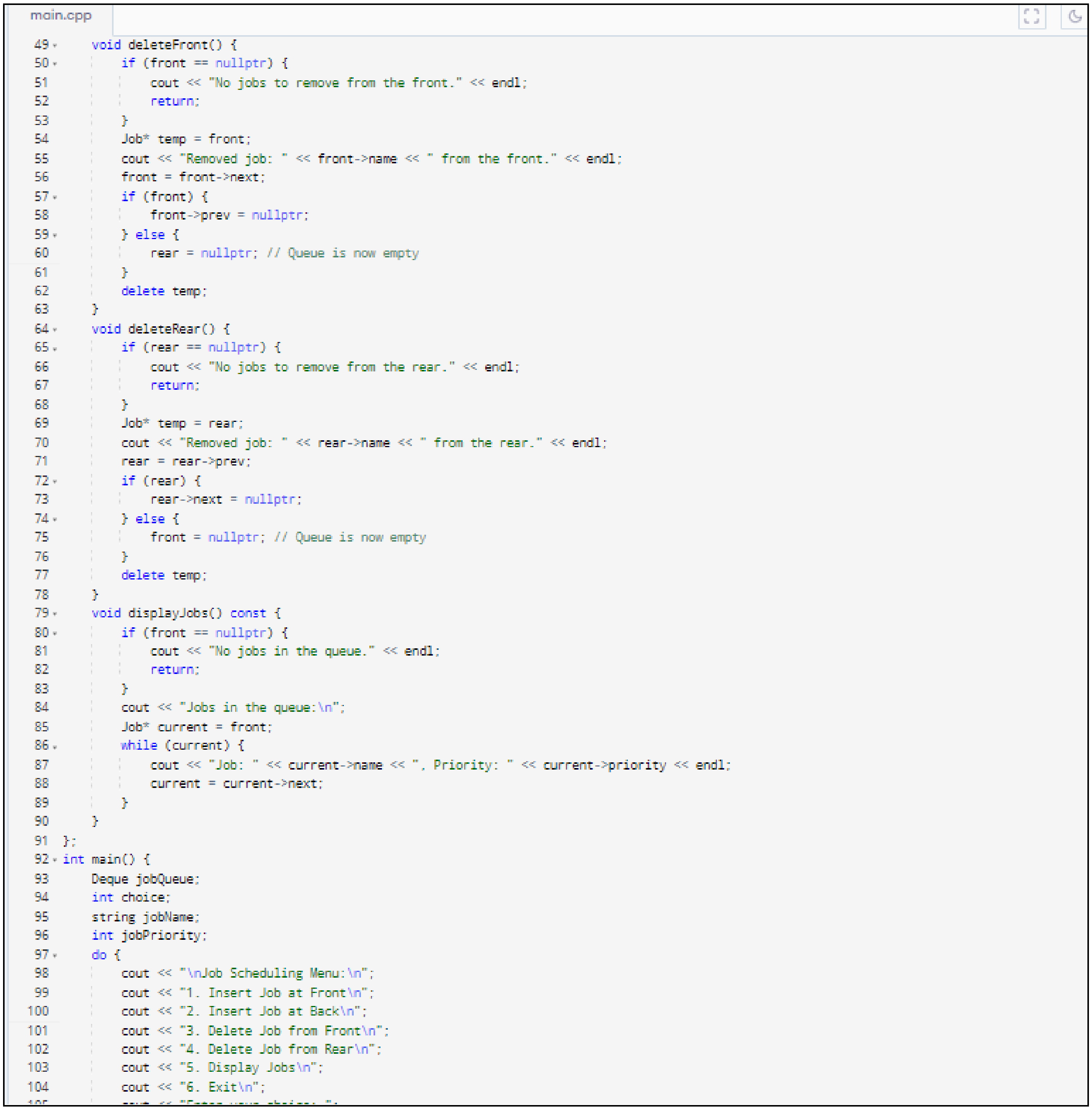
2. Insert Job at Back
3. Delete Job from Front
4. Delete Job from Rear
5. Display Jobs
6. Exit
Enter your choice:



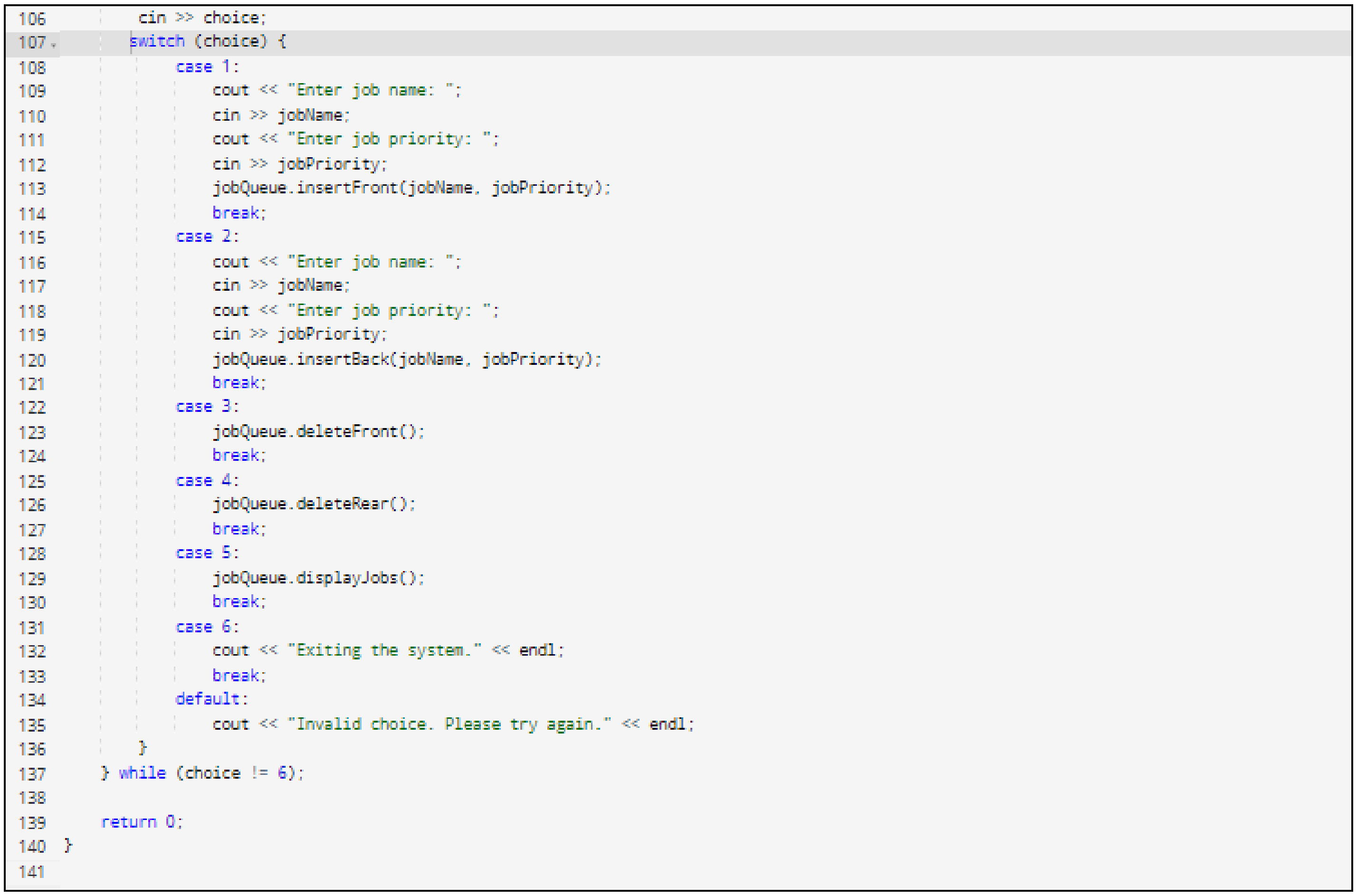
main.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;
class Job {
public:
    string name;
    int priority;
    Job* prev;
    Job* next;

    Job(string jobName, int jobPriority) {
        name = jobName;
        priority = jobPriority;
        prev = nullptr;
        next = nullptr;
    }
};
class Deque {
private:
    Job* front;
    Job* rear;
public:
    Deque() {
        front = nullptr;
        rear = nullptr;
    }
    void insertFront(string name, int priority) {
        Job* newJob = new Job(name, priority);
        if (front == nullptr) {
            front = rear = newJob;
        } else {
            newJob->next = front;
            front->prev = newJob;
            front = newJob;
        }
        cout << "Added job: " << name << " at the front." << endl;
    }
    void insertBack(string name, int priority) {
        Job* newJob = new Job(name, priority);
        if (rear == nullptr) {
            front = rear = newJob;
        } else {
            rear->next = newJob;
            newJob->prev = rear;
            rear = newJob;
        }
        cout << "Added job: " << name << " at the back." << endl;
    }
```

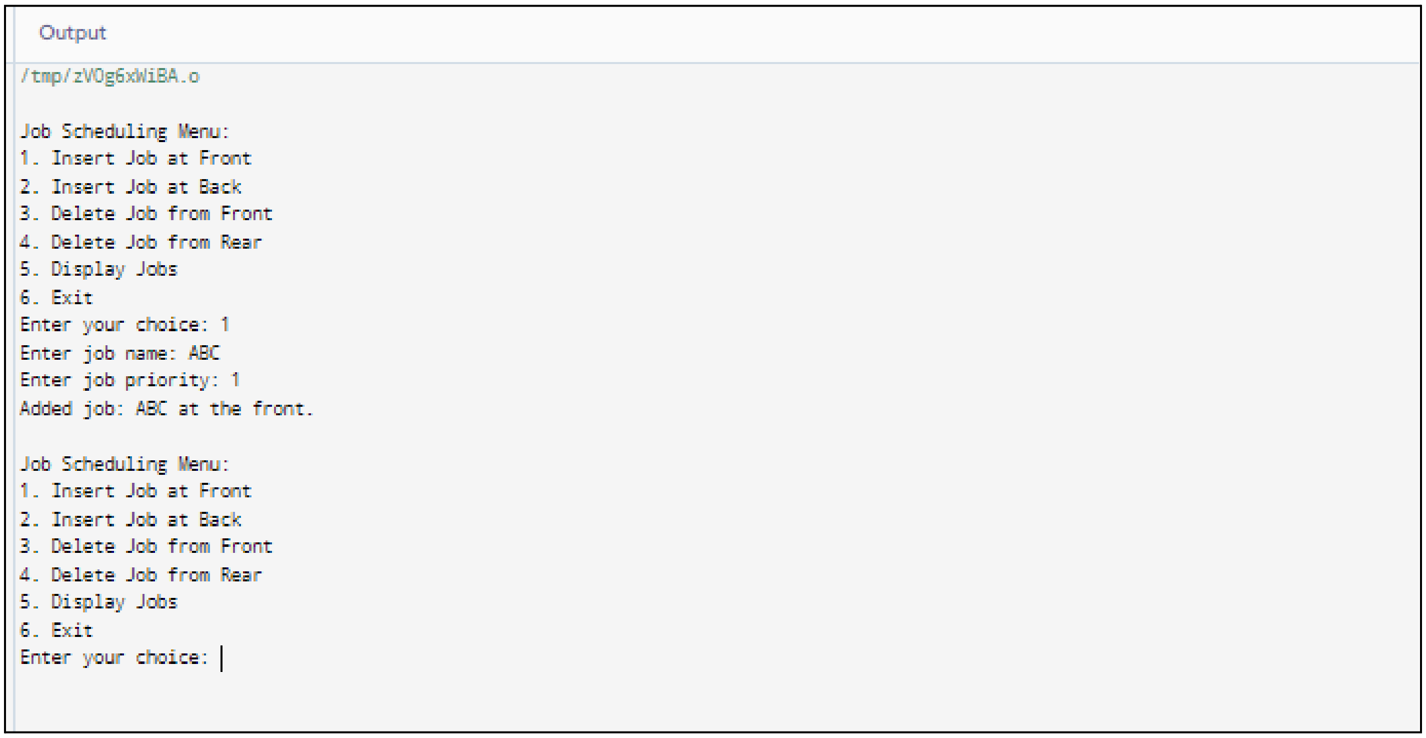

main.cpp

```cpp
    void deleteFront() {
        if (front == nullptr) {
            cout << "No jobs to remove from the front." << endl;
            return;
        }
        Job* temp = front;
        cout << "Removed job: " << front->name << " from the front." << endl;
        front = front->next;
        if (front) {
            front->prev = nullptr;
        } else {
            rear = nullptr; // Queue is now empty
        }
        delete temp;
    }
    void deleteRear() {
        if (rear == nullptr) {
            cout << "No jobs to remove from the rear." << endl;
            return;
        }
        Job* temp = rear;
        cout << "Removed job: " << rear->name << " from the rear." << endl;
        rear = rear->prev;
        if (rear) {
            rear->next = nullptr;
        } else {
            front = nullptr; // Queue is now empty
        }
        delete temp;
    }
    void displayJobs() const {
        if (front == nullptr) {
            cout << "No jobs in the queue." << endl;
            return;
        }
        cout << "Jobs in the queue:\n";
        Job* current = front;
        while (current) {
            cout << "Job: " << current->name << ", Priority: " << current->priority << endl;
            current = current->next;
        }
    }
};
int main() {
    Deque jobQueue;
    int choice;
    string jobName;
    int jobPriority;
    do {
        cout << "\nJob Scheduling Menu:\n";
        cout << "1. Insert Job at Front\n";
        cout << "2. Insert Job at Back\n";
        cout << "3. Delete Job from Front\n";
        cout << "4. Delete Job from Rear\n";
        cout << "5. Display Jobs\n";
        cout << "6. Exit\n";
```



```cpp
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Enter job name: ";
                cin >> jobName;
                cout << "Enter job priority: ";
                cin >> jobPriority;
                jobQueue.insertFront(jobName, jobPriority);
                break;
            case 2:
                cout << "Enter job name: ";
                cin >> jobName;
                cout << "Enter job priority: ";
                cin >> jobPriority;
                jobQueue.insertBack(jobName, jobPriority);
                break;
            case 3:
                jobQueue.deleteFront();
                break;
            case 4:
                jobQueue.deleteRear();
                break;
            case 5:
                jobQueue.displayJobs();
                break;
            case 6:
                cout << "Exiting the system." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 6);

    return 0;
}
```



Output

```
/tmp/zVOg6xWiBA.o

Job Scheduling Menu:
1. Insert Job at Front
2. Insert Job at Back
3. Delete Job from Front
4. Delete Job from Rear
5. Display Jobs
6. Exit
Enter your choice: 1
Enter job name: ABC
Enter job priority: 1
Added job: ABC at the front.

Job Scheduling Menu:
1. Insert Job at Front
2. Insert Job at Back
3. Delete Job from Front
4. Delete Job from Rear
5. Display Jobs
6. Exit
Enter your choice:
```