

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

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  class Job {
5  public:
6      string name;
7      int priority;
8      Job* prev;
9      Job* next;
10
11  Job(string jobName, int jobPriority) {
12      name = jobName;
13      priority = jobPriority;
14      prev = nullptr;
15      next = nullptr;
16  }
17 };
18 class Deque {
19 private:
20     Job* front;
21     Job* rear;
22 public:
23     Deque() {
24         front = nullptr;
25         rear = nullptr;
26     }
27     void insertFront(string name, int priority) {
28         Job* newJob = new Job(name, priority);
29         if (front == nullptr) {
30             front = rear = newJob;
31         } else {
32             newJob->next = front;
33             front->prev = newJob;
34             front = newJob;
35         }
36         cout << "Added job: " << name << " at the front." << endl;
37     }
38     void insertBack(string name, int priority) {
39         Job* newJob = new Job(name, priority);
40         if (rear == nullptr) {
41             front = rear = newJob;
42         } else {
43             rear->next = newJob;
44             newJob->prev = rear;
45             rear = newJob;
46         }
47         cout << "Added job: " << name << " at the back." << endl;
48     }
49 }
```

main.cpp

```
49- void deleteFront() {
50-     if (front == nullptr) {
51-         cout << "No jobs to remove from the front." << endl;
52-         return;
53-     }
54-     Job* temp = front;
55-     cout << "Removed job: " << front->name << " from the front." << endl;
56-     front = front->next;
57-     if (front) {
58-         front->prev = nullptr;
59-     } else {
60-         rear = nullptr; // Queue is now empty
61-     }
62-     delete temp;
63- }
64- void deleteRear() {
65-     if (rear == nullptr) {
66-         cout << "No jobs to remove from the rear." << endl;
67-         return;
68-     }
69-     Job* temp = rear;
70-     cout << "Removed job: " << rear->name << " from the rear." << endl;
71-     rear = rear->prev;
72-     if (rear) {
73-         rear->next = nullptr;
74-     } else {
75-         front = nullptr; // Queue is now empty
76-     }
77-     delete temp;
78- }
79- void displayJobs() const {
80-     if (front == nullptr) {
81-         cout << "No jobs in the queue." << endl;
82-         return;
83-     }
84-     cout << "Jobs in the queue:\n";
85-     Job* current = front;
86-     while (current) {
87-         cout << "Job: " << current->name << ", Priority: " << current->priority << endl;
88-         current = current->next;
89-     }
90- }
91- };
92- int main() {
93-     Deque jobQueue;
94-     int choice;
95-     string jobName;
96-     int jobPriority;
97-     do {
98-         cout << "\nJob Scheduling Menu:\n";
99-         cout << "1. Insert Job at Front\n";
100-         cout << "2. Insert Job at Back\n";
101-         cout << "3. Delete Job from Front\n";
102-         cout << "4. Delete Job from Rear\n";
103-         cout << "5. Display Jobs\n";
104-         cout << "6. Exit\n";
105-         cout << "Enter your choice: ";
```

```

106     cin >> choice;
107     switch (choice) {
108     case 1:
109         cout << "Enter job name: ";
110         cin >> jobName;
111         cout << "Enter job priority: ";
112         cin >> jobPriority;
113         jobQueue.insertFront(jobName, jobPriority);
114         break;
115     case 2:
116         cout << "Enter job name: ";
117         cin >> jobName;
118         cout << "Enter job priority: ";
119         cin >> jobPriority;
120         jobQueue.insertBack(jobName, jobPriority);
121         break;
122     case 3:
123         jobQueue.deleteFront();
124         break;
125     case 4:
126         jobQueue.deleteRear();
127         break;
128     case 5:
129         jobQueue.displayJobs();
130         break;
131     case 6:
132         cout << "Exiting the system." << endl;
133         break;
134     default:
135         cout << "Invalid choice. Please try again." << endl;
136     }
137 } while (choice != 6);
138
139 return 0;
140 }
141

```

Output

/tmp/zV0g6xWiBA.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: |