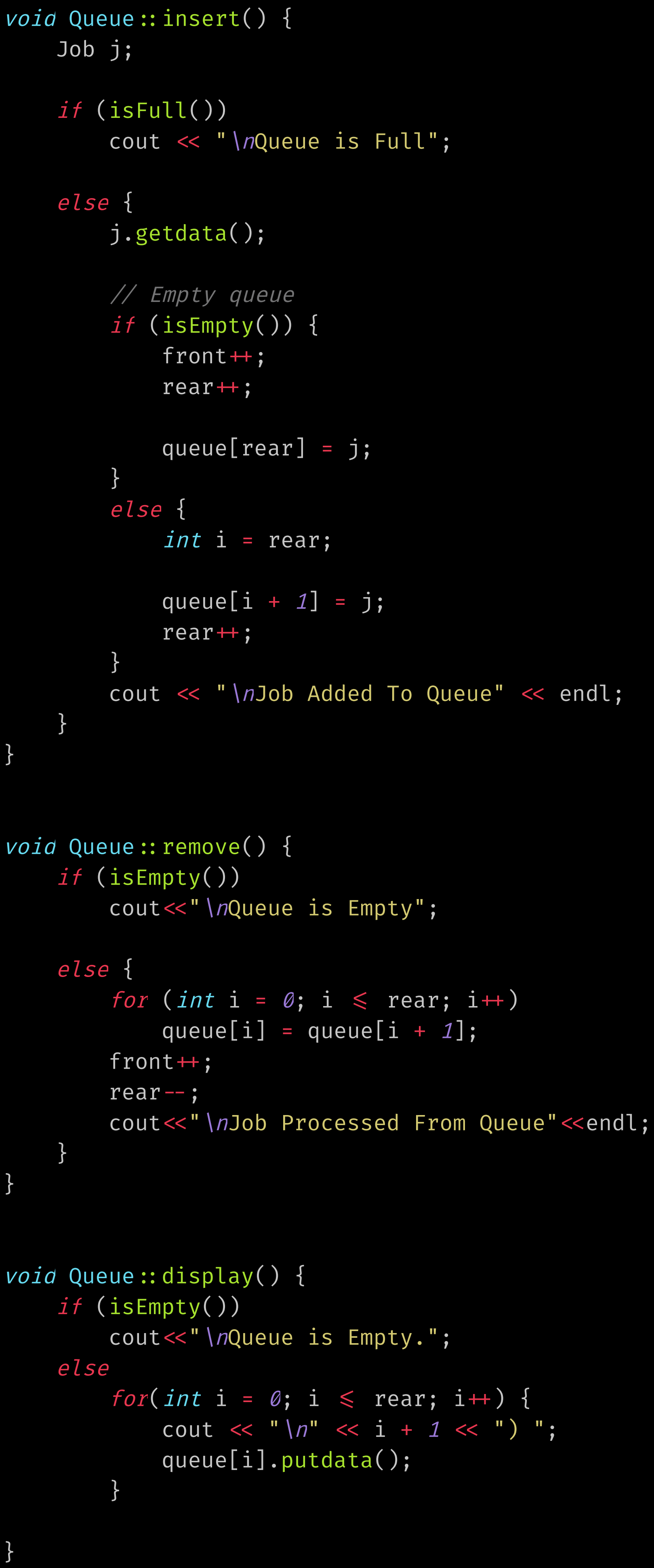
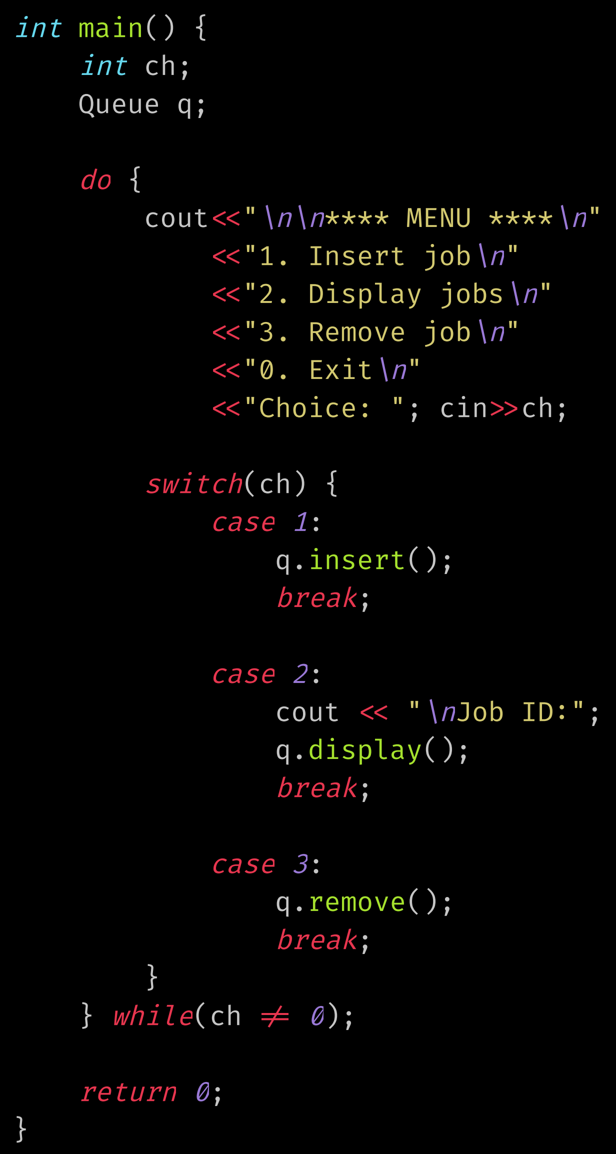
```cpp
/* DSL - EXPERIMENT 11 - E29 */

#include<iostream>
using namespace std;

#define MAX 5


class Job {
    int ID;
    friend class Queue;        //Queue can access private members of class Job

    public:
        void getdata() {
            cout << "\nEnter Job ID: "; cin >> ID;
        }
        void putdata() {
            cout << 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 == (rear + 1) || rear == -1);
}


bool Queue::isFull() {
    return (rear == (MAX - 1));
}
```



```cpp
void Queue::insert() {
    Job j;

    if (isFull())
        cout << "\nQueue is Full";

    else {
        j.getdata();

        // Empty queue
        if (isEmpty()) {
            front++;
            rear++;

            queue[rear] = j;
        }
        else {
            int i = rear;

            queue[i + 1] = j;
            rear++;
        }
        cout << "\nJob Added To Queue" << endl;
    }
}


void Queue::remove() {
    if (isEmpty())
        cout<<"\nQueue is Empty";

    else {
        for (int i = 0; i <= rear; i++)
            queue[i] = queue[i + 1];
        front++;
        rear--;
        cout<<"\nJob Processed From Queue"<<endl;
    }
}


void Queue::display() {
    if (isEmpty())
        cout<<"\nQueue is Empty.";
    else
        for(int i = 0; i <= rear; i++) {
            cout << "\n" << i + 1 << ") ";
            queue[i].putdata();
        }

}
```



```cpp
int main() {
    int ch;
    Queue q;

    do {
        cout<<"\n\n**** MENU ****\n"
            <<"1. Insert job\n"
            <<"2. Display jobs\n"
            <<"3. Remove job\n"
            <<"0. Exit\n"
            <<"Choice: "; cin>>ch;

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

            case 2:
                cout << "\nJob ID:";
                q.display();
                break;

            case 3:
                q.remove();
                break;
        }
    } while(ch != 0);

    return 0;
}
```

```
/*

----------------- OUTPUT ------------------

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

Enter Job ID: 8546

Job Added To Queue


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

Enter Job ID: 452

Job Added To Queue


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

Enter Job ID: 33524

Job Added To Queue


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

Enter Job ID: 2457

Job Added To Queue
```

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

Enter Job ID: 4196

Job Added To Queue


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

Queue is Full

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

Job ID:
1) 8546
2) 452
3) 33524
4) 2457
5) 4196

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

Job Processed From Queue


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

```
Job ID:
1) 452
2) 33524
3) 2457
4) 4196

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

Job Processed From Queue


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

Job ID:
1) 33524
2) 2457
3) 4196

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

Enter Job ID: 832

Job Added To Queue


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

Job ID:
1) 33524
2) 2457
```

```
3) 4196
4) 832

**** MENU ****
1. Insert job
2. Display jobs
3. Remove job
0. Exit
Choice: 0

*/
```