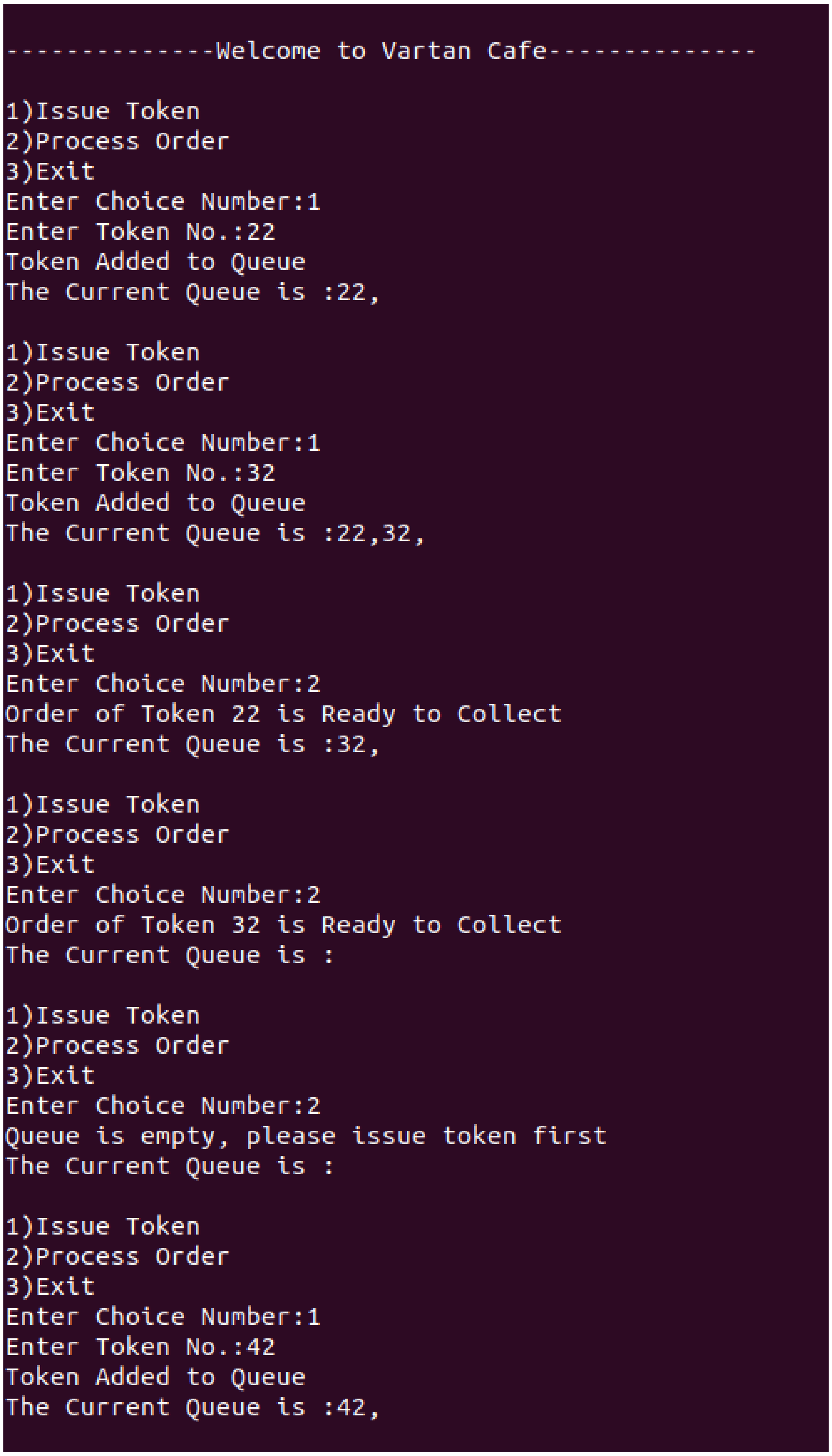
DS TUTORIAL 6

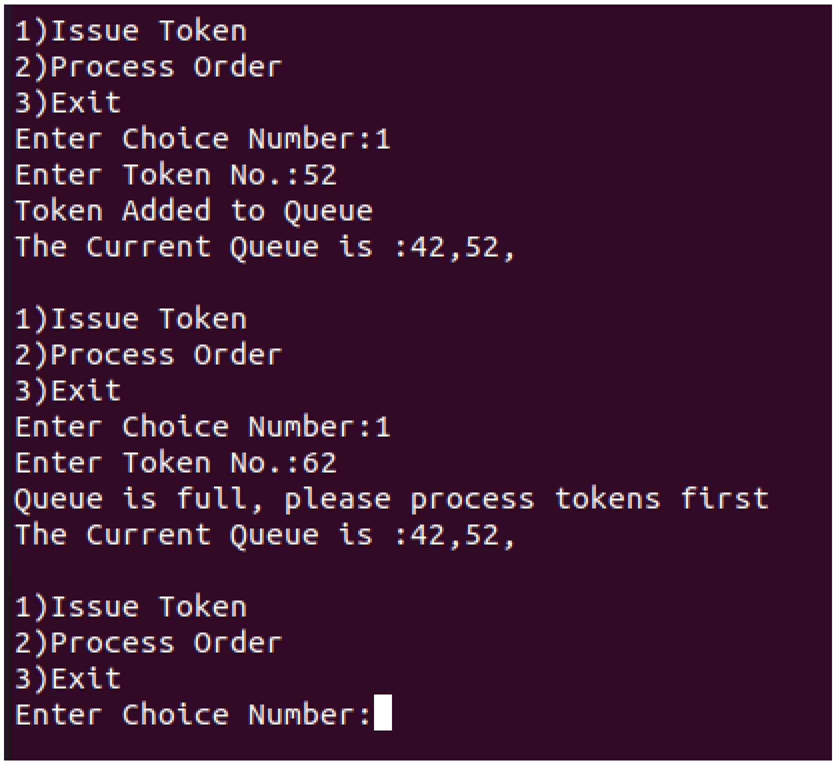
CODE 1:

```
/*Coffee Shop Line (Simple Queue):
Arrival: Customers arrive at the coffee shop and stand in line. Order
Processing: The first customer in line gets their order taken, and
the barista starts making the coffee.Serving: Once the first customer
is served, they leave the queue, and the next customer in line moves
forward to be served. Write a program to implement a simple queue*/

#include <iostream>
using namespace std;
class Coffee{
private:
    int rear=-1,front=0,maxlen=4,token[4];
public:
    void Enqueue(int t);
    void Dequeue();
    bool isEmpty();
    bool isFull();
    void display();
};
void Coffee::Dequeue(){
    if(isEmpty()){
        cout<<"Order of Token "<<token[front]<<" is Ready to
Collect\n";
        token[front++]=0;
    };
};
void Coffee::Enqueue(int t){
    if(isFull()){
        token[++rear]=t;
        cout<<"Token Added to Queue\n";
    };
};
bool Coffee::isFull(){
    if(rear==maxlen-1){
        cout<<"Queue is full, please process tokens first\n";
        return 0;
    }
    else{
        return 1;
    };
};
bool Coffee::isEmpty(){
    if(rear==-1 || front>rear){
        cout<<"Queue is empty, please issue token first\n";
        return 0;
    }
    else{
        return 1;
    };
};
void Coffee::display(){
     cout<<"The Current Queue is :";
     for(int i=front;i<=rear;i++){
          cout<<token[i]<<",";

     }
}
int main(){
    Coffee shop;
    int choice=0,tnumber;
    while(choice!=3){
        cout<<"\n--------------Welcome to
Starbucks--------------"<<endl;
        cout<<"1)Issue Token\n";
        cout<<"2)Process Order\n";
        cout<<"3)Exit\n";
        cout<<"Enter Choice Number:";
        cin>>choice;
        if(choice==0){
            cout<<"Please Enter Choice Correctly";
            break;
        }
        else if(choice==1){
            cout<<"Enter Token No.:";
            cin>>tnumber;
            shop.Enqueue(tnumber);
            shop.display();
        }
        else if(choice==2){
            shop.Dequeue();
            shop.display();
        };
    };
    cout<<"Gooddbye,Come Again!!!!!!!\n";
    return 0;
};
```

OUTPUT 1



```
--------------Welcome to Vartan Cafe--------------

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:1
Enter Token No.:22
Token Added to Queue
The Current Queue is :22,

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:1
Enter Token No.:32
Token Added to Queue
The Current Queue is :22,32,

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:2
Order of Token 22 is Ready to Collect
The Current Queue is :32,

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:2
Order of Token 32 is Ready to Collect
The Current Queue is :

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:2
Queue is empty, please issue token first
The Current Queue is :

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:1
Enter Token No.:42
Token Added to Queue
The Current Queue is :42,
```

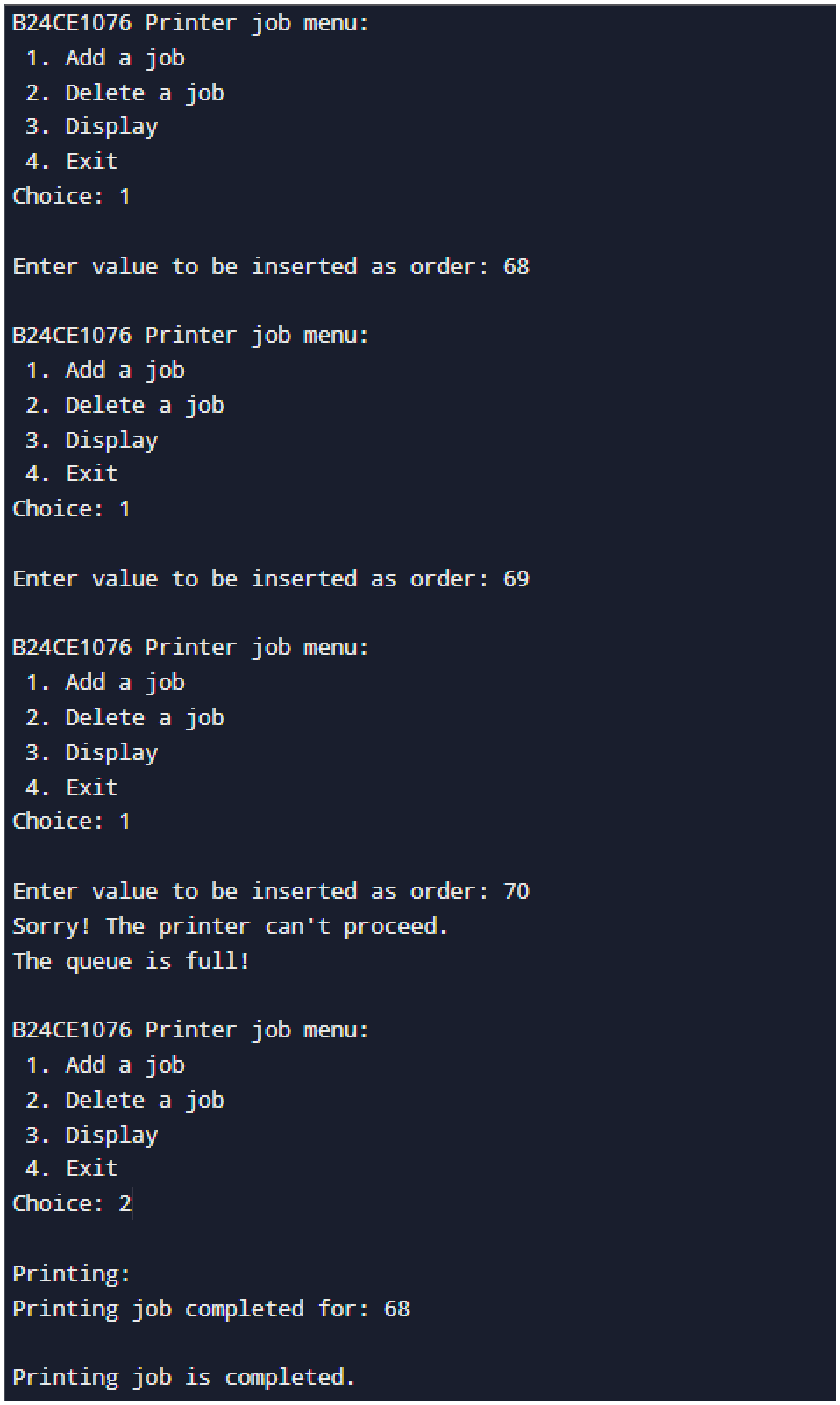

```
1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:1
Enter Token No.:52
Token Added to Queue
The Current Queue is :42,52,

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:1
Enter Token No.:62
Queue is full, please process tokens first
The Current Queue is :42,52,

1)Issue Token
2)Process Order
3)Exit
Enter Choice Number:
```

CODE 2:

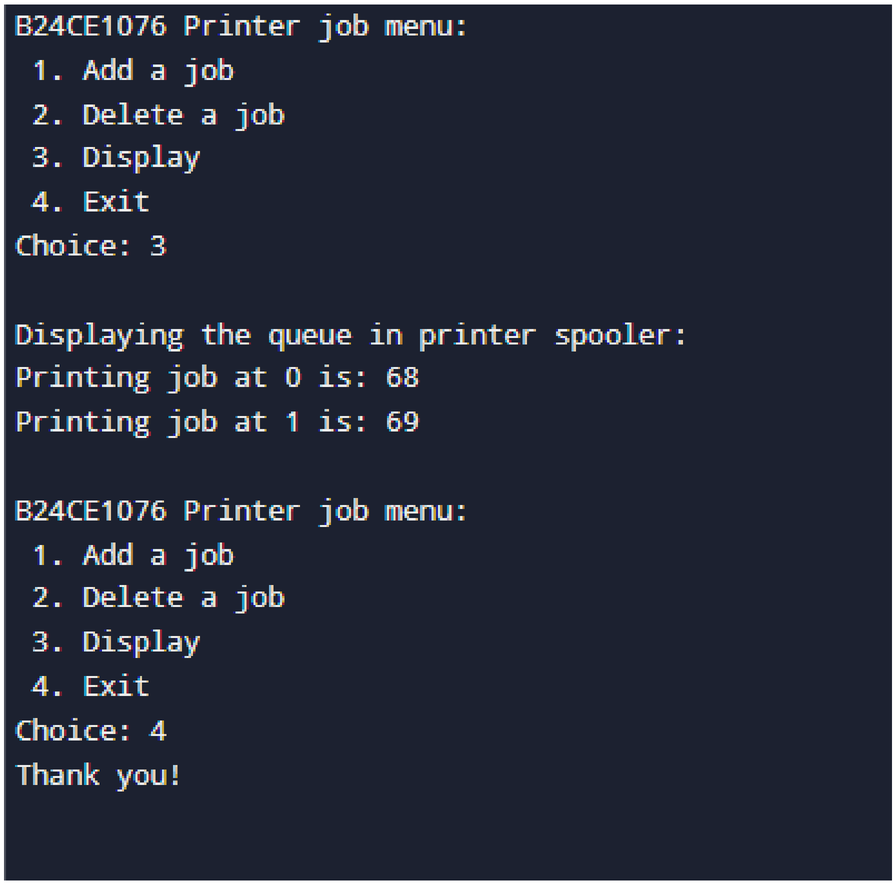
```
/*Printer Spooler (Circular Queue): In a multi-user environment,
printers often use a circular queue to manage print jobs. Each
print job is added to the queue, and the printer processes them
in the order they arrive. Once a print job is completed, it
moves out of the queue, and the next job is processed,
efficiently managing the flow of print tasks. Implement the
Printer Spooler system using a circular queue without using
built-in queues. */
#include <iostream>
#define MAXSIZE 2
#define MIN 0
using namespace std;

class Printer_spooler {
    int token[MAXSIZE];
public:
    int rear;
    int front;
    Printer_spooler() {
        rear = -1;
        front = -1;
    }
    bool isFull();
    bool isEmpty();
    void enQueue(int t);
    int deQueue();
    void display();
};

// Check if queue is full (Circular Queue Condition)
bool Printer_spooler::isFull() {
    if (((rear + 1) % MAXSIZE) == front) {
        return 1;
    } else {
        return 0;
    }
}

// Check if queue is empty
bool Printer_spooler::isEmpty() {
    if (front == -1) {
        return 1;
    } else {
        return 0;
    }
}

// Add a job to the printer queue
void Printer_spooler::enQueue(int t) {
    if (isFull()) {
        cout << "Sorry! The printer can't proceed.\nThe queue is
full!" << endl;
    } else {
        if (rear == -1) {
            front = 0;
        }
        rear = (rear + 1) % MAXSIZE;
        token[rear] = t;
    }
}

// Remove a job from the printer queue (simulate printing)
```

```
int Printer_spooler::deQueue() {
    if (isEmpty()) {
        cout << "The queue is empty." << endl;
        return 0;
    } else if (front == rear) {
        int t = token[front];
        cout << "Printing job completed for: " << t << endl;
        front = -1;
        rear = -1;
        return t;
    } else {
        int t = token[front];
        cout << "Printing job completed for: " << t << endl;
        front = (front + 1) % MAXSIZE;
        return t;
    }
}

// Display all jobs in the printer queue
void Printer_spooler::display() {
    cout << "\nDisplaying the queue in printer spooler:" <<
endl;
    for (int i = 0; i < MAXSIZE; i++) {
        cout << "Printing job at " << i << " is: " << token[i]
<< endl;
    }
}

int main() {
    Printer_spooler customer;
    int choice = 0, order;
    do {
        cout << "B24CE1076 Printer job menu:" << endl;
        cout << " 1. Add a job\n 2. Delete a job \n 3. Display\n
4. Exit\nChoice: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                cout << "\nEnter value to be inserted as order:
";
                cin >> order;
                customer.enQueue(order);
                cout << endl;
                break;
            }
            case 2: {
                cout << "\nPrinting:" << endl;
                customer.deQueue();
                cout << "\nPrinting job is completed." << endl;
                break;
            }
            case 3: {
                customer.display();
                cout << endl;
                break;
            }
            case 4: {
                cout << "Thank you!" << endl;
                break;
            }
            default: {
                cout << "\nIncorrect choice! Try again" << endl;
            }
        }
    } while (choice != 4);
    return 0;
}
```

OUTPUT 2:



```
B24CE1076 Printer job menu:
 1. Add a job
 2. Delete a job
 3. Display
 4. Exit
Choice: 1

Enter value to be inserted as order: 68

B24CE1076 Printer job menu:
 1. Add a job
 2. Delete a job
 3. Display
 4. Exit
Choice: 1

Enter value to be inserted as order: 69

B24CE1076 Printer job menu:
 1. Add a job
 2. Delete a job
 3. Display
 4. Exit
Choice: 1

Enter value to be inserted as order: 70
Sorry! The printer can't proceed.
The queue is full!

B24CE1076 Printer job menu:
 1. Add a job
 2. Delete a job
 3. Display
 4. Exit
Choice: 2

Printing:
Printing job completed for: 68

Printing job is completed.
```



```
B24CE1076 Printer job menu:
 1. Add a job
 2. Delete a job
 3. Display
 4. Exit
Choice: 3

Displaying the queue in printer spooler:
Printing job at 0 is: 68
Printing job at 1 is: 69

B24CE1076 Printer job menu:
 1. Add a job
 2. Delete a job
 3. Display
 4. Exit
Choice: 4
Thank you!
```