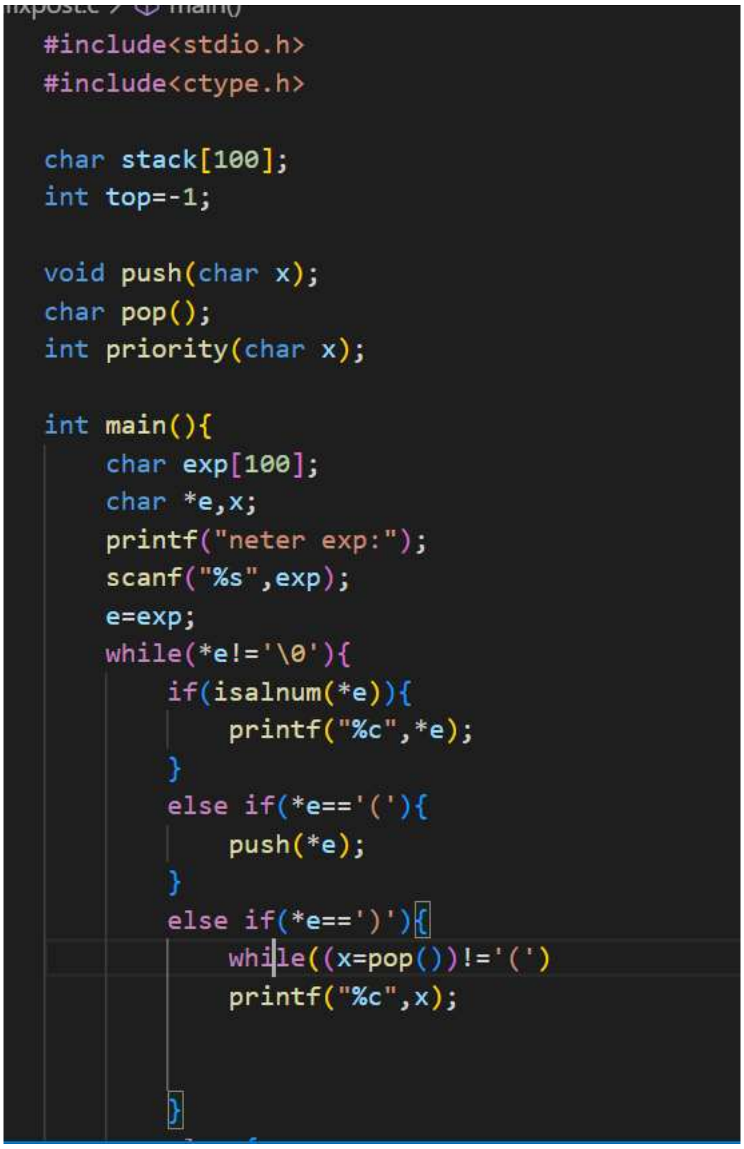
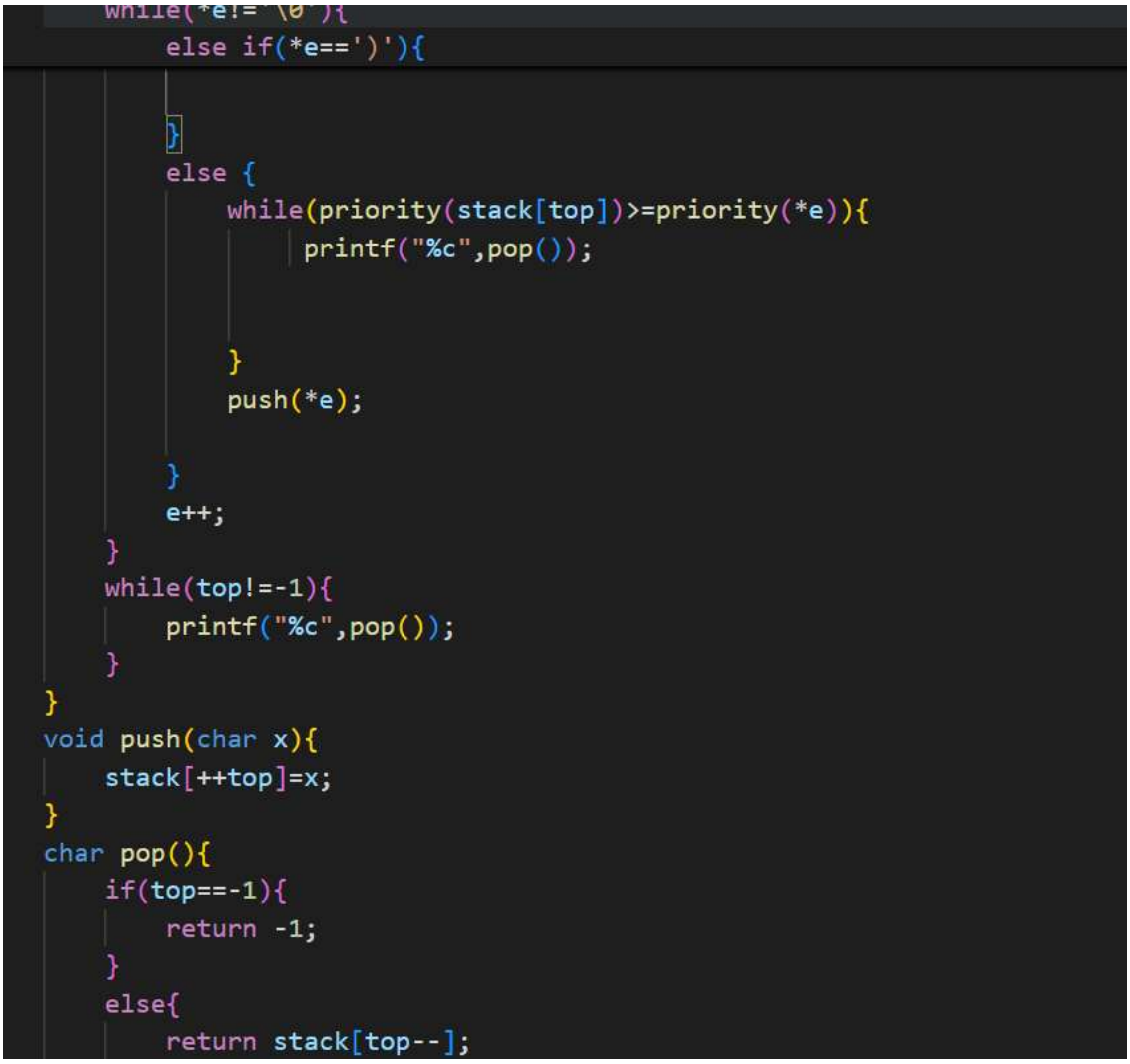
DAY 23-DAILY ASSIGNMENTS

ANSU MARIUM SHIBU

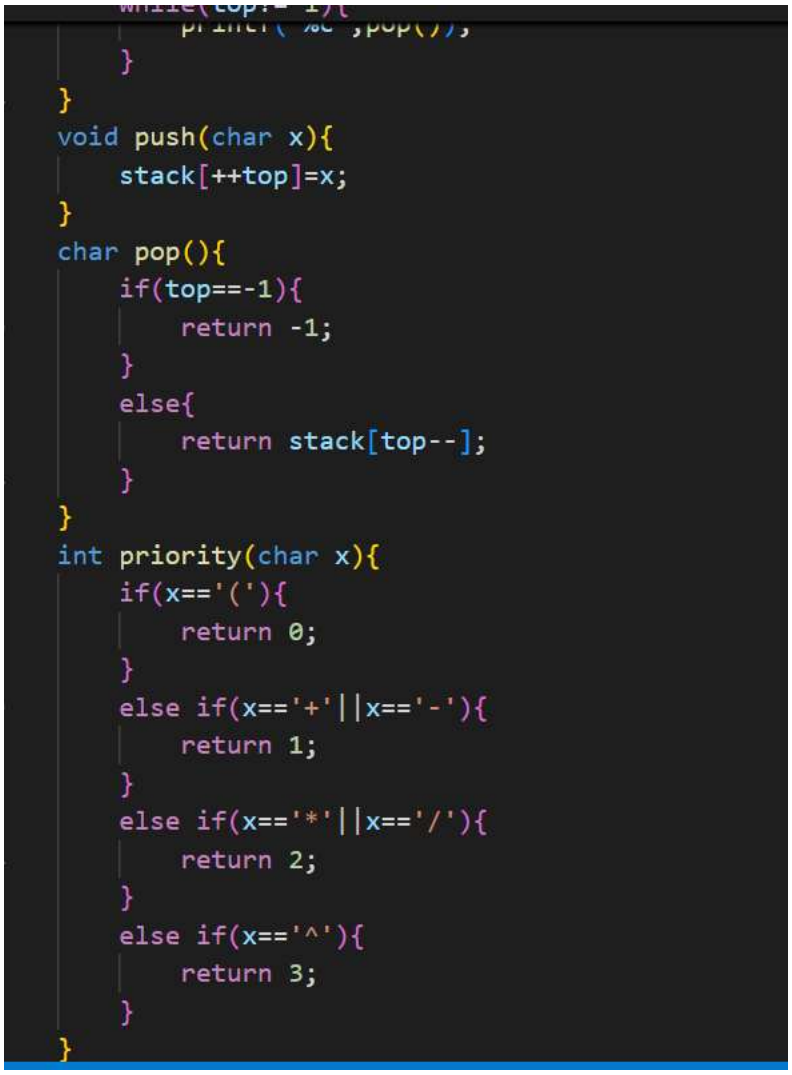
1.infix to postfix



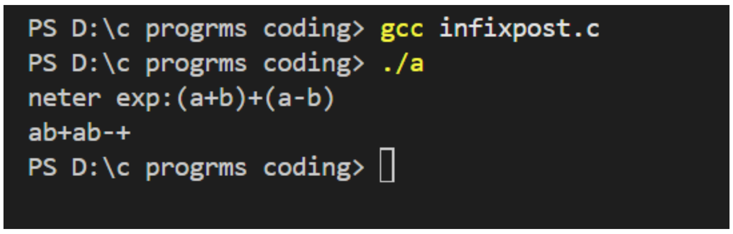
```
#include<stdio.h>
#include<ctype.h>

char stack[100];
int top=-1;

void push(char x);
char pop();
int priority(char x);

int main(){
    char exp[100];
    char *e,x;
    printf("neter exp:");
    scanf("%s",exp);
    e=exp;
    while(*e!='\0'){
        if(isalnum(*e)){
            printf("%c",*e);
        }
        else if(*e=='('){
            push(*e);
        }
        else if(*e==')'){
            while((x=pop())!='(')
            printf("%c",x);


        }
```



```
    while(*e!='\0'){
        else if(*e==')'){

        }
        else {
            while(priority(stack[top])>=priority(*e)){
                printf("%c",pop());


            }
            push(*e);

        }
        e++;
    }
    while(top!=-1){
        printf("%c",pop());
    }
}
void push(char x){
    stack[++top]=x;
}
char pop(){
    if(top==-1){
        return -1;
    }
    else{
        return stack[top--];
```

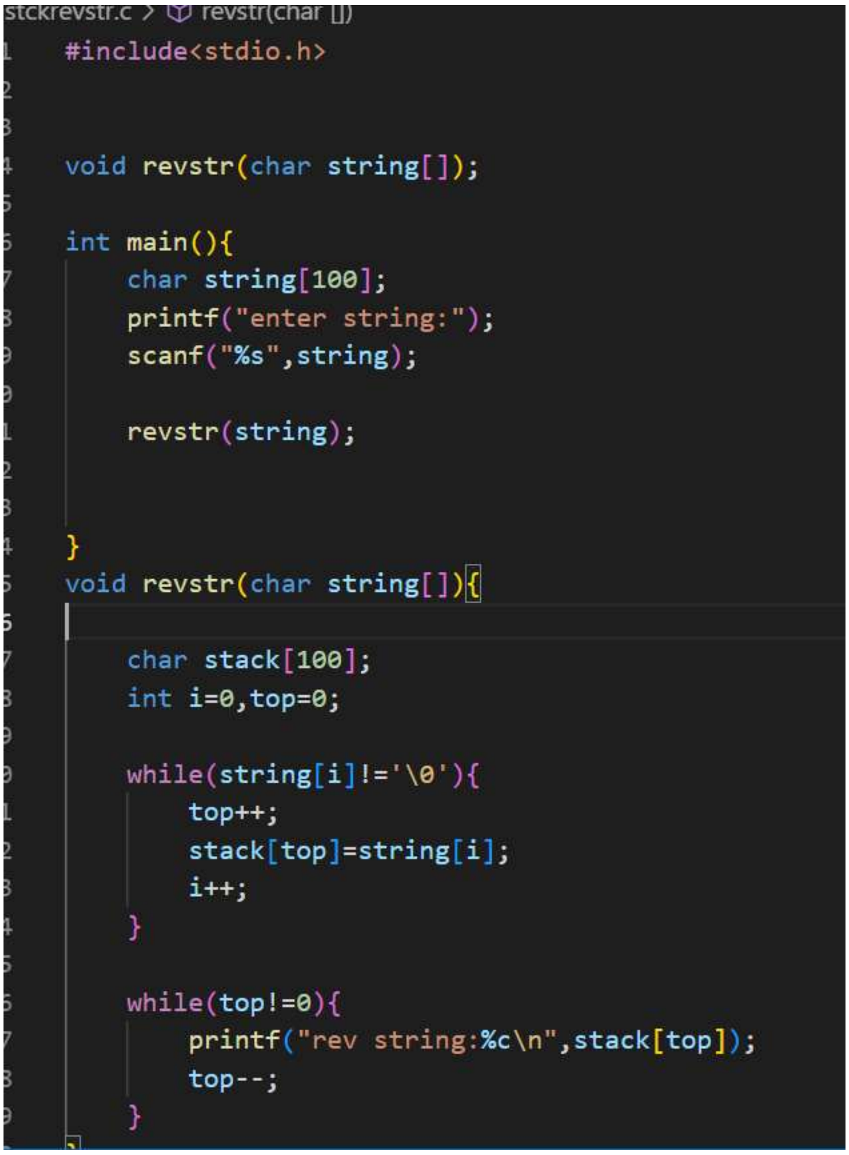


```c
    while(top!=-1){
        printf("%c",pop());
    }
}
void push(char x){
    stack[++top]=x;
}
char pop(){
    if(top==-1){
        return -1;
    }
    else{
        return stack[top--];
    }
}
int priority(char x){
    if(x=='('){
        return 0;
    }
    else if(x=='+'||x=='-'){
        return 1;
    }
    else if(x=='*'||x=='/'){
        return 2;
    }
    else if(x=='^'){
        return 3;
    }
}
```

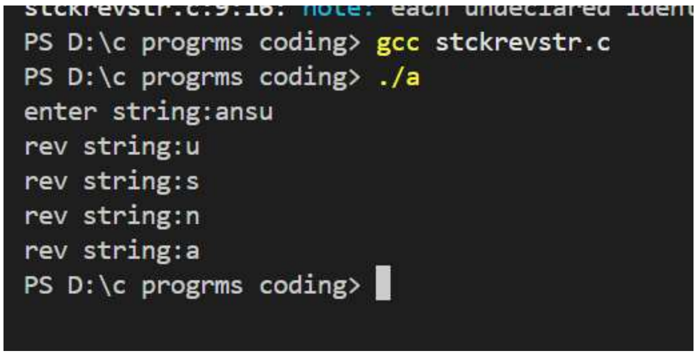


```
PS D:\c progrms coding> gcc infixpost.c
PS D:\c progrms coding> ./a
neter exp:(a+b)+(a-b)
ab+ab-+
PS D:\c progrms coding> 
```

2.Reverse a string using stack



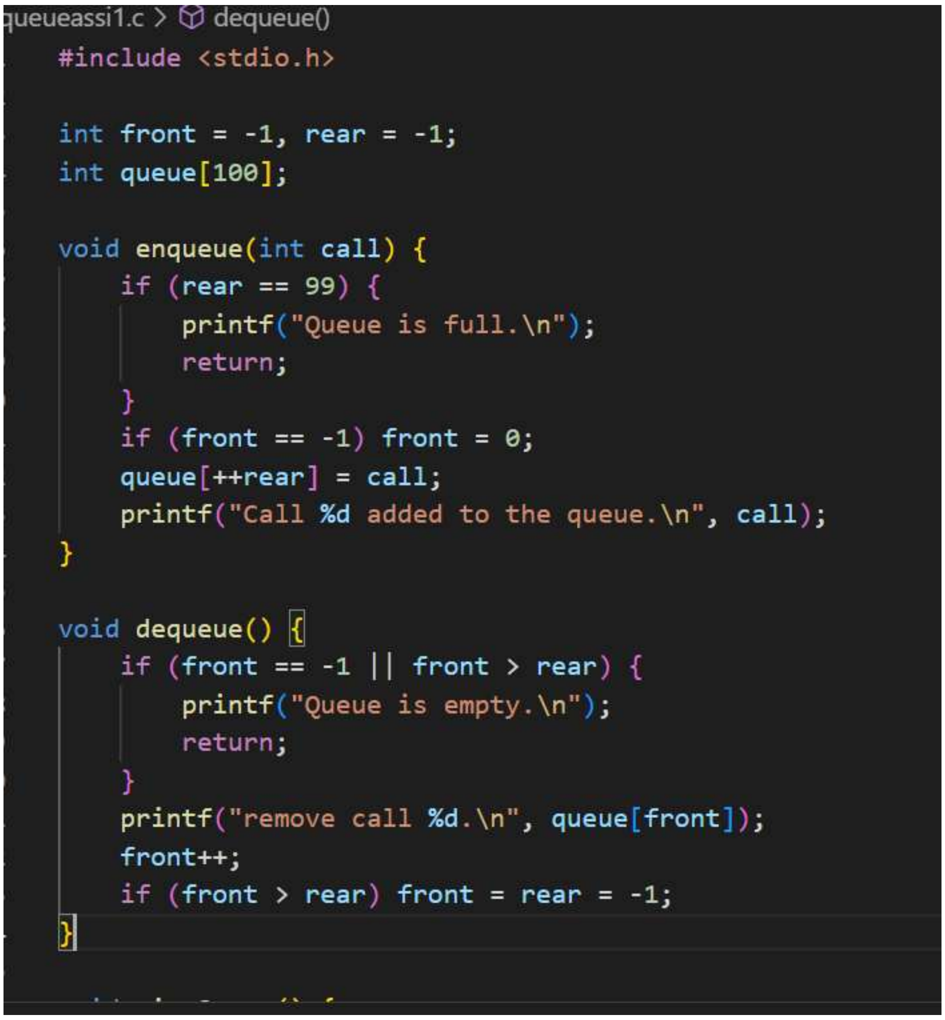
```c
stckrevstr.c > revstr(char [])
#include<stdio.h>


void revstr(char string[]);

int main(){
    char string[100];
    printf("enter string:");
    scanf("%s",string);

    revstr(string);


}
void revstr(char string[]){

    char stack[100];
    int i=0,top=0;

    while(string[i]!='\0'){
        top++;
        stack[top]=string[i];
        i++;
    }

    while(top!=0){
        printf("rev string:%c\n",stack[top]);
        top--;
    }
```



```
stckrevstr.c:9:16: note: each undeclared ident
PS D:\c progrms coding> gcc stckrevstr.c
PS D:\c progrms coding> ./a
enter string:ansu
rev string:u
rev string:s
rev string:n
rev string:a
PS D:\c progrms coding> 
```

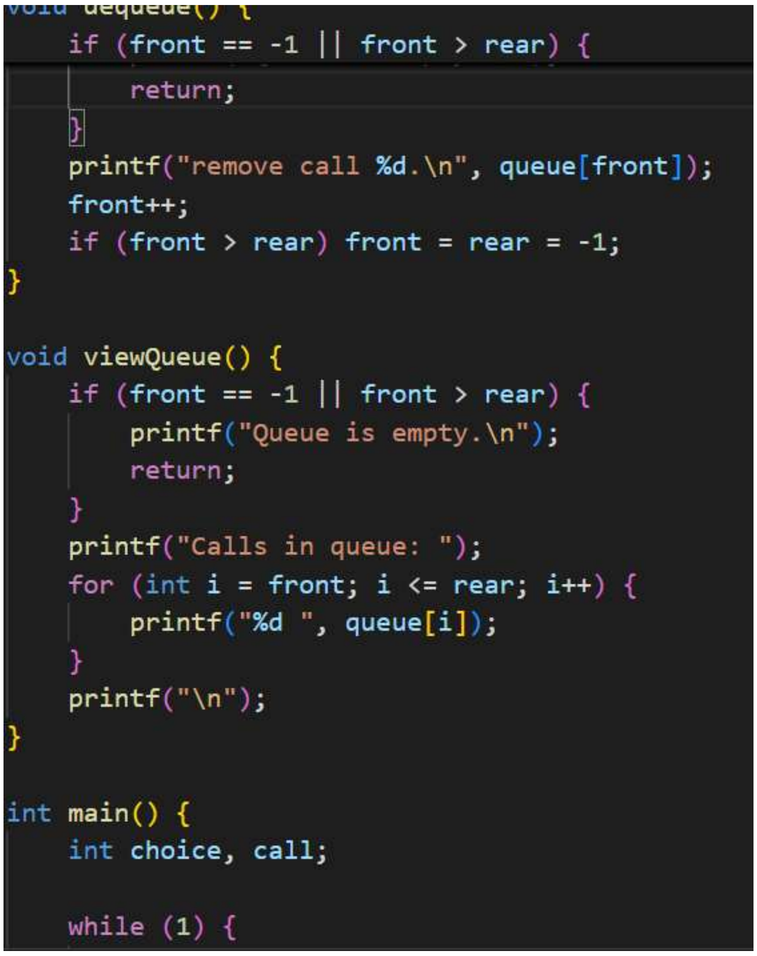
3. **1.Simulate a Call Center Queue**

Create a program to simulate a call center where incoming calls are handled on a first-come, first-served basis. Use a queue to manage call handling and provide options to add, remove, and view calls

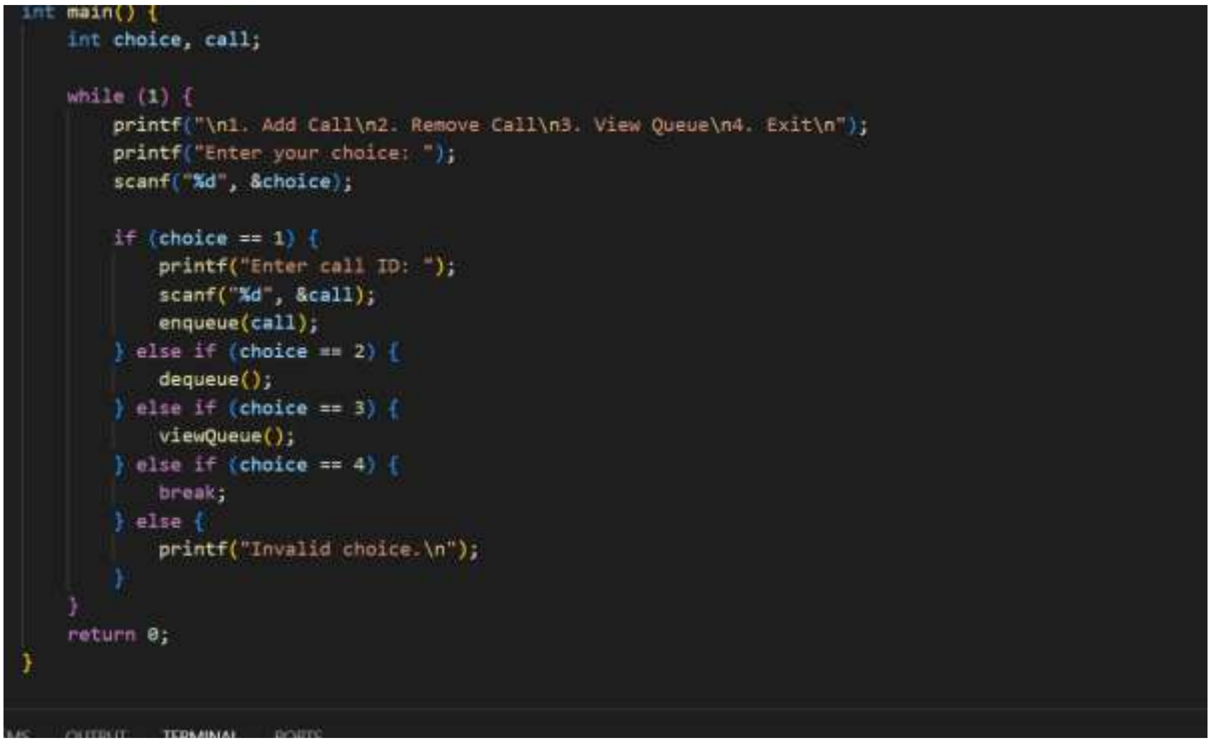


queueassi1.c > dequeue()

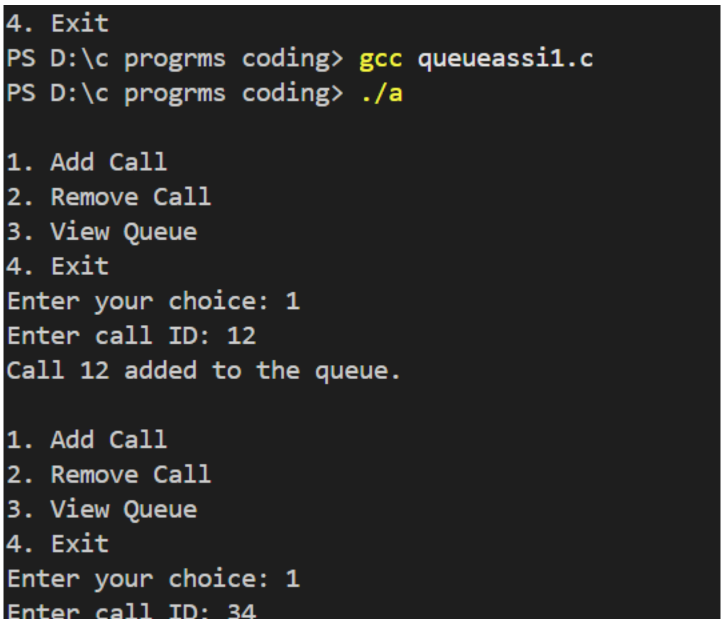
```c
#include <stdio.h>

int front = -1, rear = -1;
int queue[100];

void enqueue(int call) {
    if (rear == 99) {
        printf("Queue is full.\n");
        return;
    }
    if (front == -1) front = 0;
    queue[++rear] = call;
    printf("Call %d added to the queue.\n", call);
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
        return;
    }
    printf("remove call %d.\n", queue[front]);
    front++;
    if (front > rear) front = rear = -1;
}
```



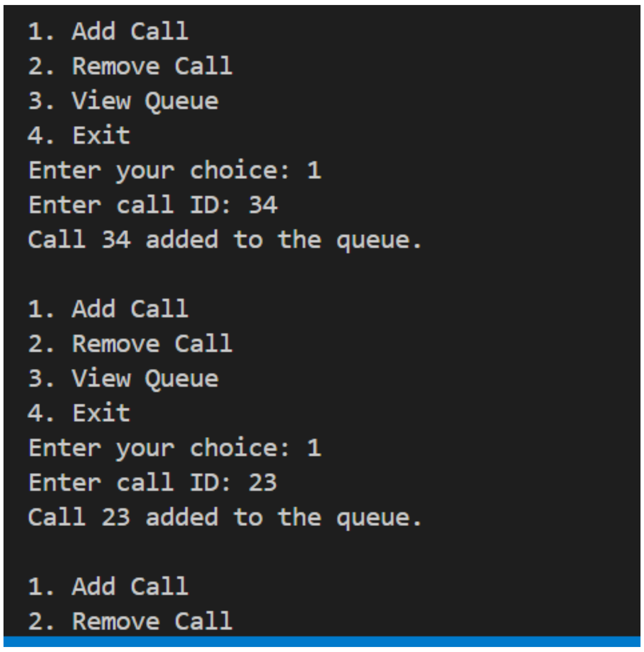
```c
void dequeue() {
    if (front == -1 || front > rear) {
        return;
    }
    printf("remove call %d.\n", queue[front]);
    front++;
    if (front > rear) front = rear = -1;
}

void viewQueue() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Calls in queue: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int choice, call;

    while (1) {
```



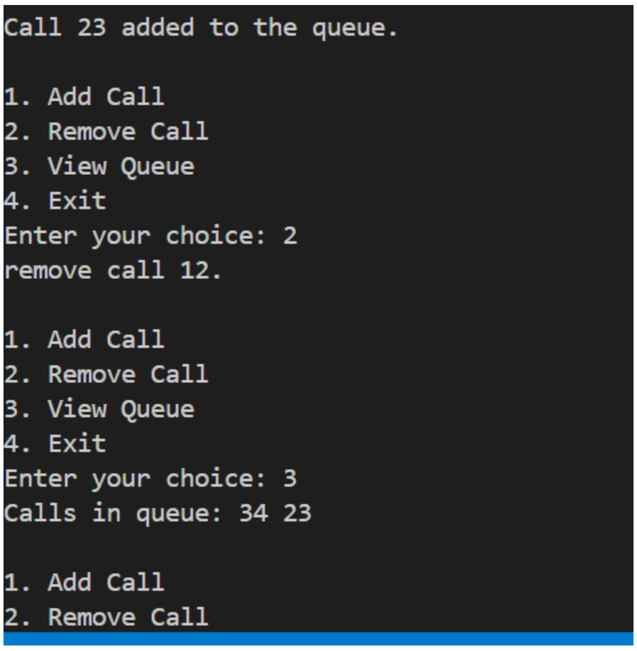
```c
int main() {
    int choice, call;

    while (1) {
        printf("\n1. Add Call\n2. Remove Call\n3. View Queue\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            printf("Enter call ID: ");
            scanf("%d", &call);
            enqueue(call);
        } else if (choice == 2) {
            dequeue();
        } else if (choice == 3) {
            viewQueue();
        } else if (choice == 4) {
            break;
        } else {
            printf("Invalid choice.\n");
        }
    }
    return 0;
}
```



```
4. Exit
PS D:\c progrms coding> gcc queueassi1.c
PS D:\c progrms coding> ./a

1. Add Call
2. Remove Call
3. View Queue
4. Exit
Enter your choice: 1
Enter call ID: 12
Call 12 added to the queue.

1. Add Call
2. Remove Call
3. View Queue
4. Exit
Enter your choice: 1
Enter call ID: 34
```



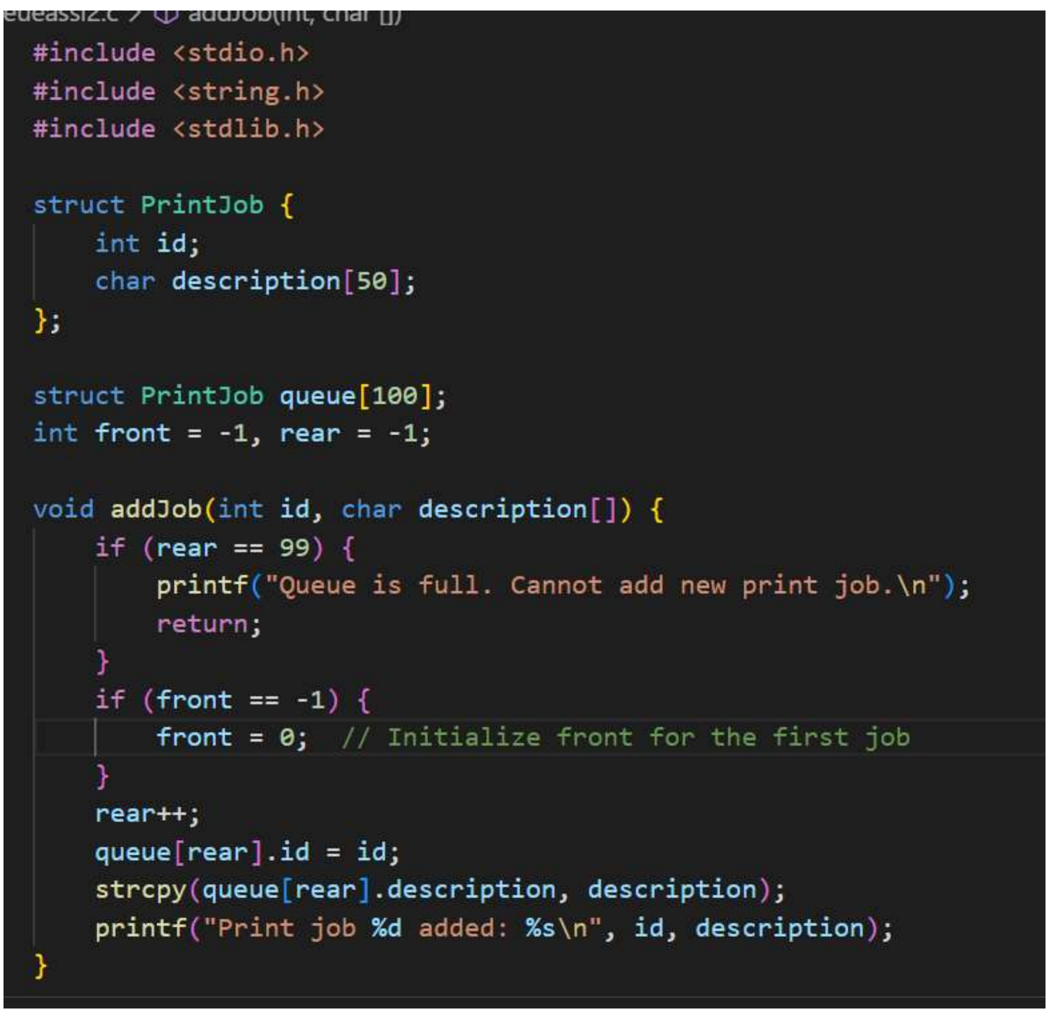
```
1. Add Call
2. Remove Call
3. View Queue
4. Exit
Enter your choice: 1
Enter call ID: 34
Call 34 added to the queue.

1. Add Call
2. Remove Call
3. View Queue
4. Exit
Enter your choice: 1
Enter call ID: 23
Call 23 added to the queue.

1. Add Call
2. Remove Call
```



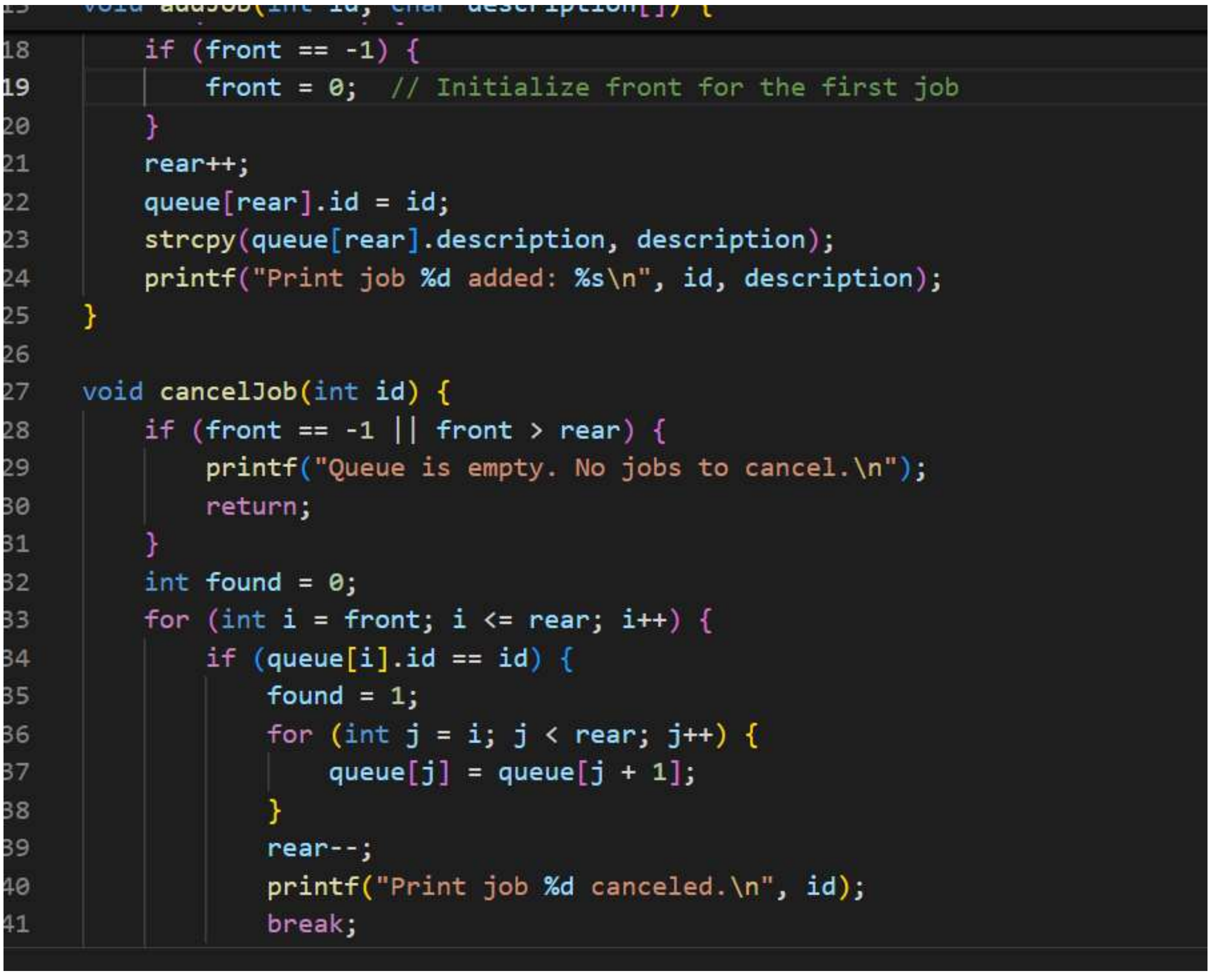
```
Call 23 added to the queue.

1. Add Call
2. Remove Call
3. View Queue
4. Exit
Enter your choice: 2
remove call 12.

1. Add Call
2. Remove Call
3. View Queue
4. Exit
Enter your choice: 3
Calls in queue: 34 23

1. Add Call
2. Remove Call
```

4. 2.Print Job Scheduler

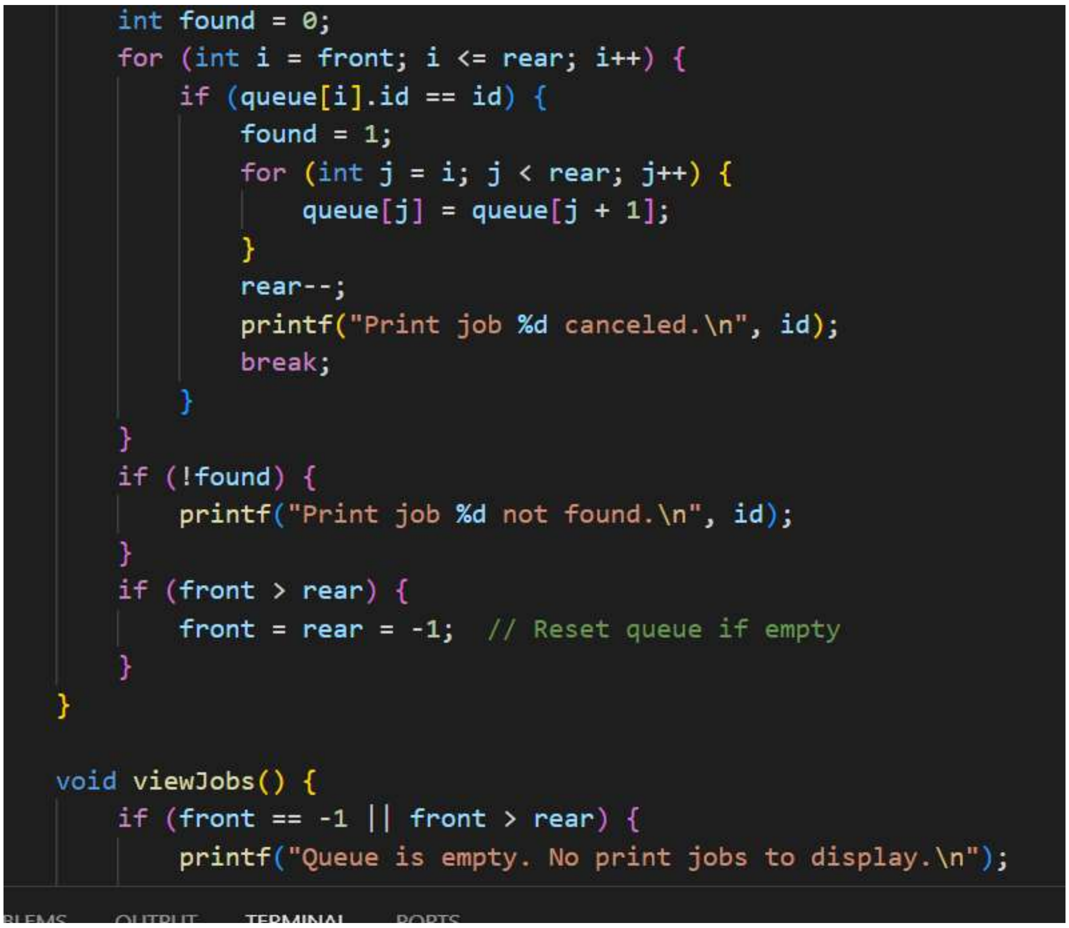
Implement a print job scheduler where print requests are queued. Allow users to add new print jobs, cancel a specific job, and print jobs in the order they were added.



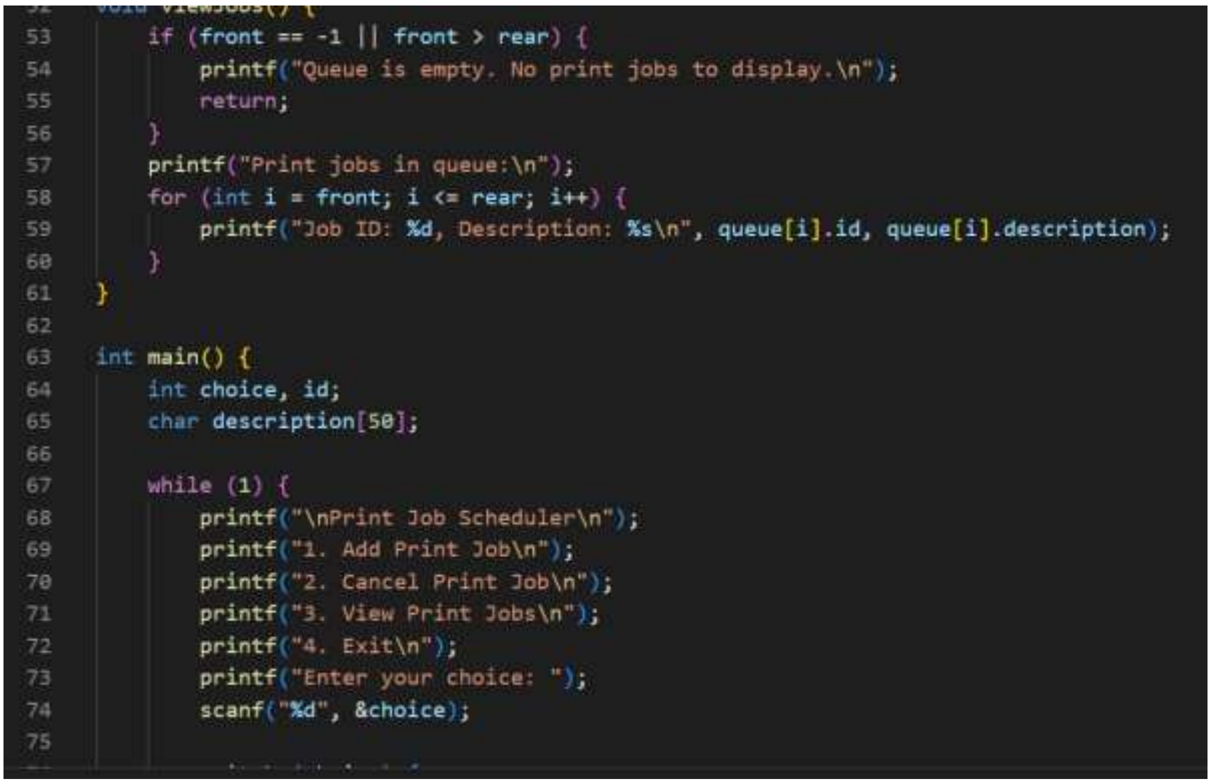
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct PrintJob {
    int id;
    char description[50];
};

struct PrintJob queue[100];
int front = -1, rear = -1;

void addJob(int id, char description[]) {
    if (rear == 99) {
        printf("Queue is full. Cannot add new print job.\n");
        return;
    }
    if (front == -1) {
        front = 0;  // Initialize front for the first job
    }
    rear++;
    queue[rear].id = id;
    strcpy(queue[rear].description, description);
    printf("Print job %d added: %s\n", id, description);
}
```



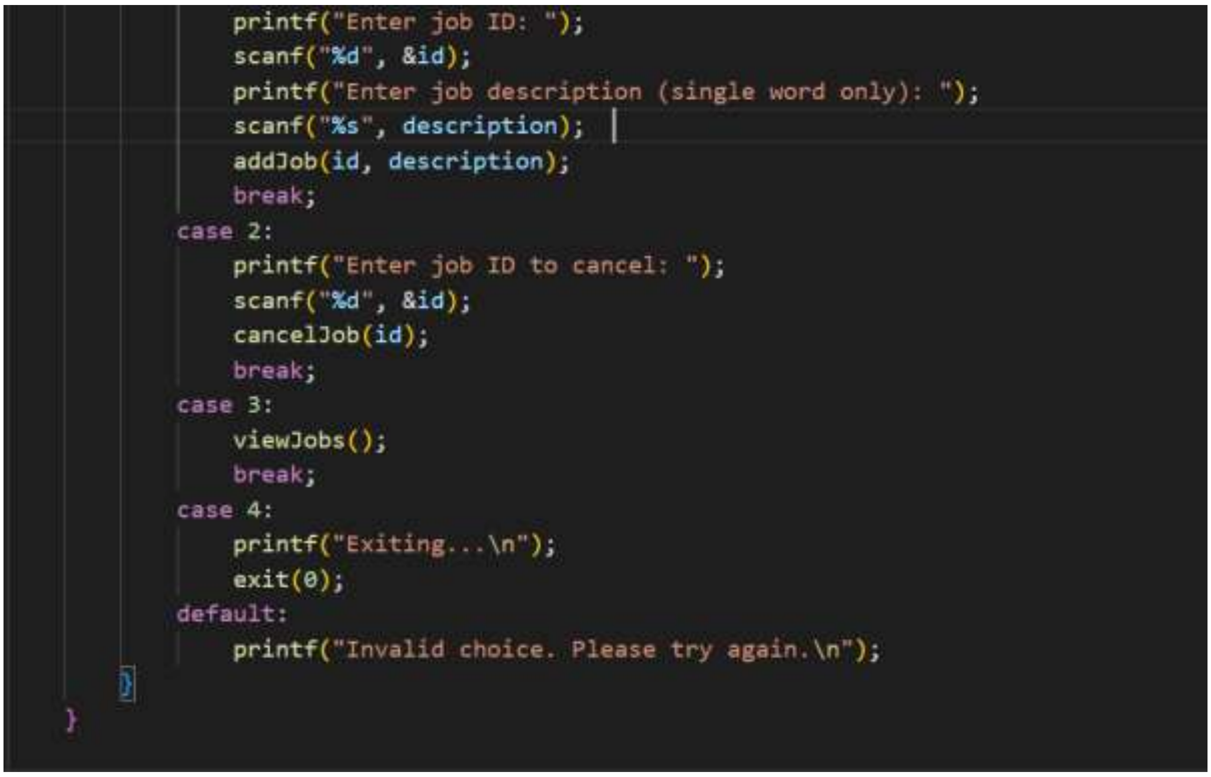
```c
void addJob(int id, char description[]) {
    if (front == -1) {
        front = 0;  // Initialize front for the first job
    }
    rear++;
    queue[rear].id = id;
    strcpy(queue[rear].description, description);
    printf("Print job %d added: %s\n", id, description);
}

void cancelJob(int id) {
    if (front == -1 || front > rear) {
        printf("Queue is empty. No jobs to cancel.\n");
        return;
    }
    int found = 0;
    for (int i = front; i <= rear; i++) {
        if (queue[i].id == id) {
            found = 1;
            for (int j = i; j < rear; j++) {
                queue[j] = queue[j + 1];
            }
            rear--;
            printf("Print job %d canceled.\n", id);
            break;
```



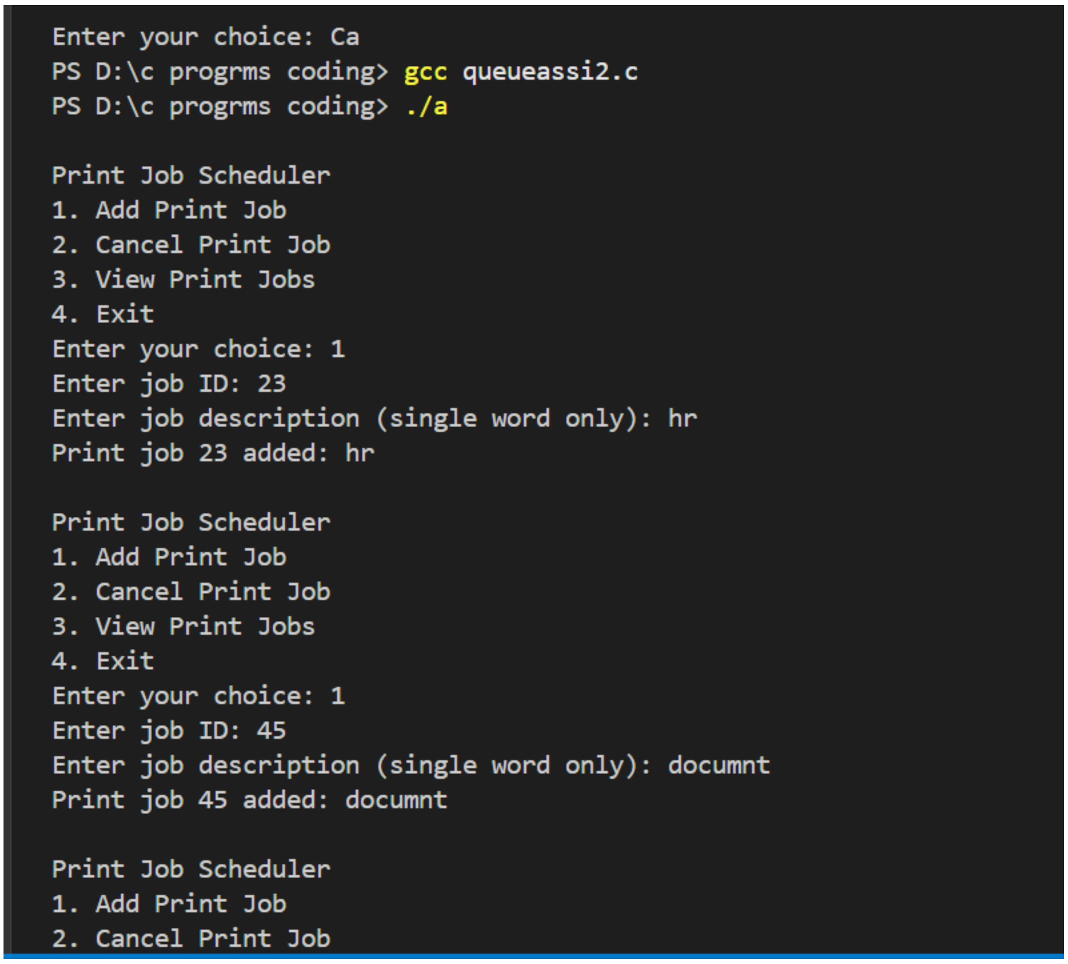
```c
    int found = 0;
    for (int i = front; i <= rear; i++) {
        if (queue[i].id == id) {
            found = 1;
            for (int j = i; j < rear; j++) {
                queue[j] = queue[j + 1];
            }
            rear--;
            printf("Print job %d canceled.\n", id);
            break;
        }
    }
    if (!found) {
        printf("Print job %d not found.\n", id);
    }
    if (front > rear) {
        front = rear = -1;  // Reset queue if empty
    }
}

void viewJobs() {
    if (front == -1 || front > rear) {
        printf("Queue is empty. No print jobs to display.\n");
```



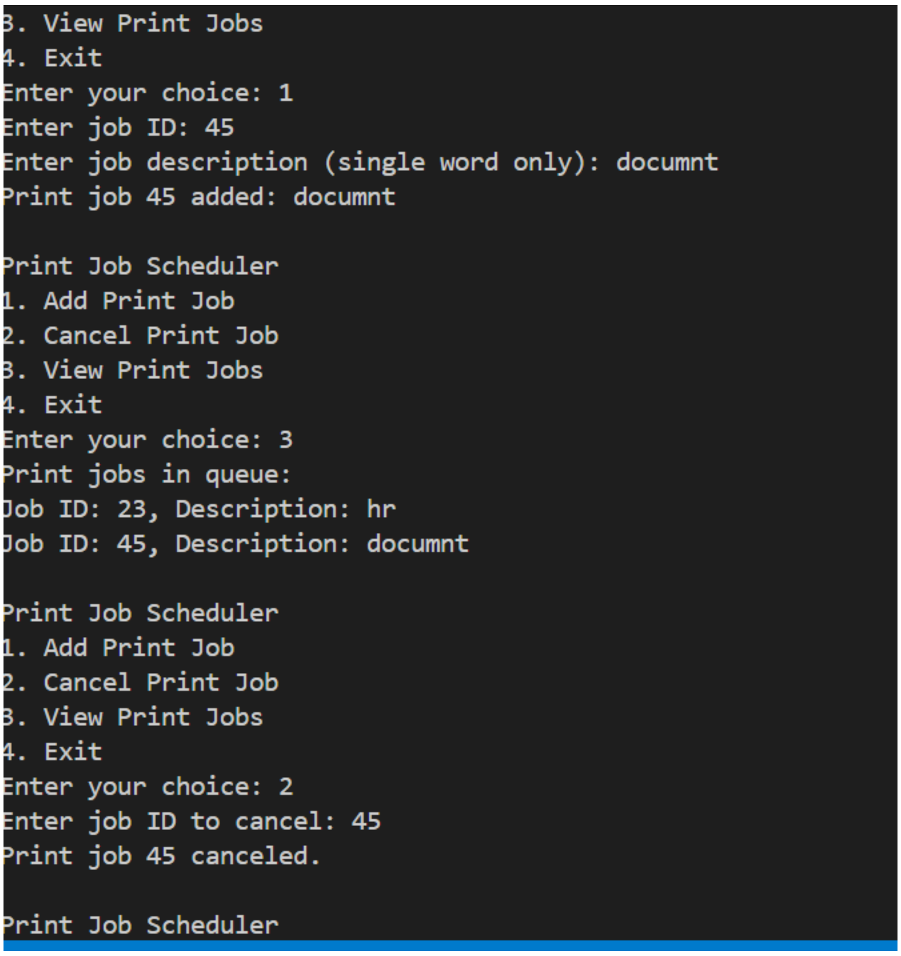
```c
    if (front == -1 || front > rear) {
        printf("Queue is empty. No print jobs to display.\n");
        return;
    }
    printf("Print jobs in queue:\n");
    for (int i = front; i <= rear; i++) {
        printf("Job ID: %d, Description: %s\n", queue[i].id, queue[i].description);
    }
}

int main() {
    int choice, id;
    char description[50];

    while (1) {
        printf("\nPrint Job Scheduler\n");
        printf("1. Add Print Job\n");
        printf("2. Cancel Print Job\n");
        printf("3. View Print Jobs\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

```



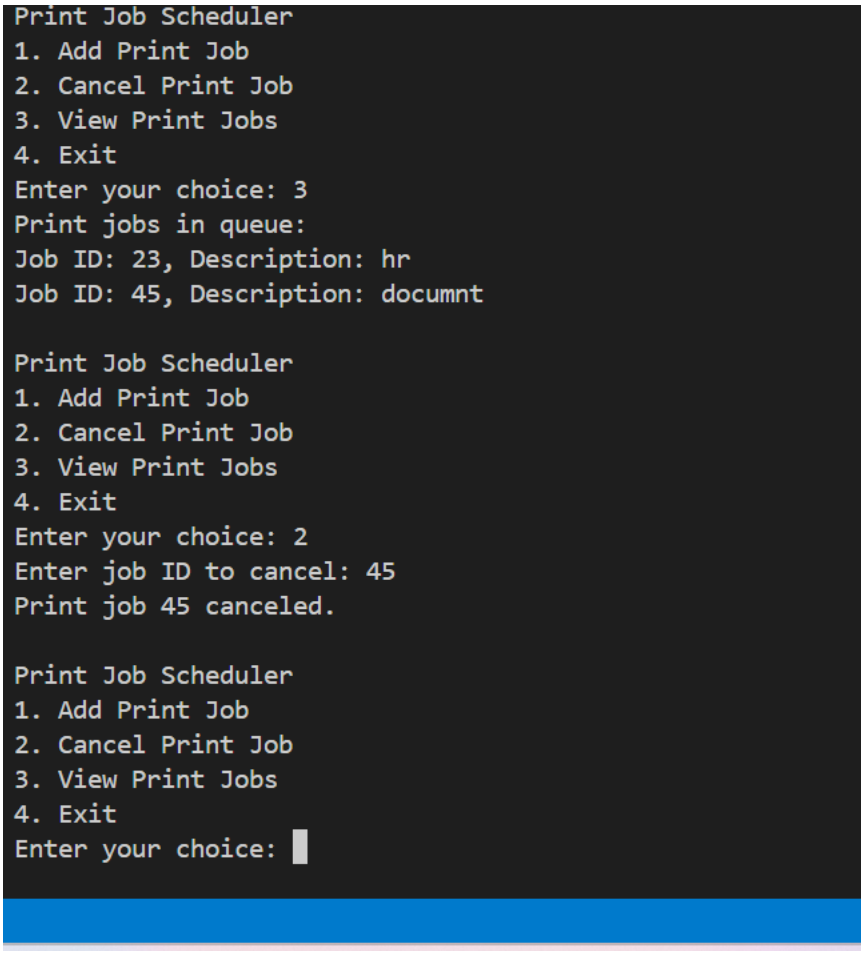
```
                printf("Enter job ID: ");
                scanf("%d", &id);
                printf("Enter job description (single word only): ");
                scanf("%s", description);
                addJob(id, description);
                break;
            case 2:
                printf("Enter job ID to cancel: ");
                scanf("%d", &id);
                cancelJob(id);
                break;
            case 3:
                viewJobs();
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
```



```
Enter your choice: Ca
PS D:\c progrms coding> gcc queueassi2.c
PS D:\c progrms coding> ./a

Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
3. View Print Jobs
4. Exit
Enter your choice: 1
Enter job ID: 23
Enter job description (single word only): hr
Print job 23 added: hr

Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
3. View Print Jobs
4. Exit
Enter your choice: 1
Enter job ID: 45
Enter job description (single word only): documnt
Print job 45 added: documnt

Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
```



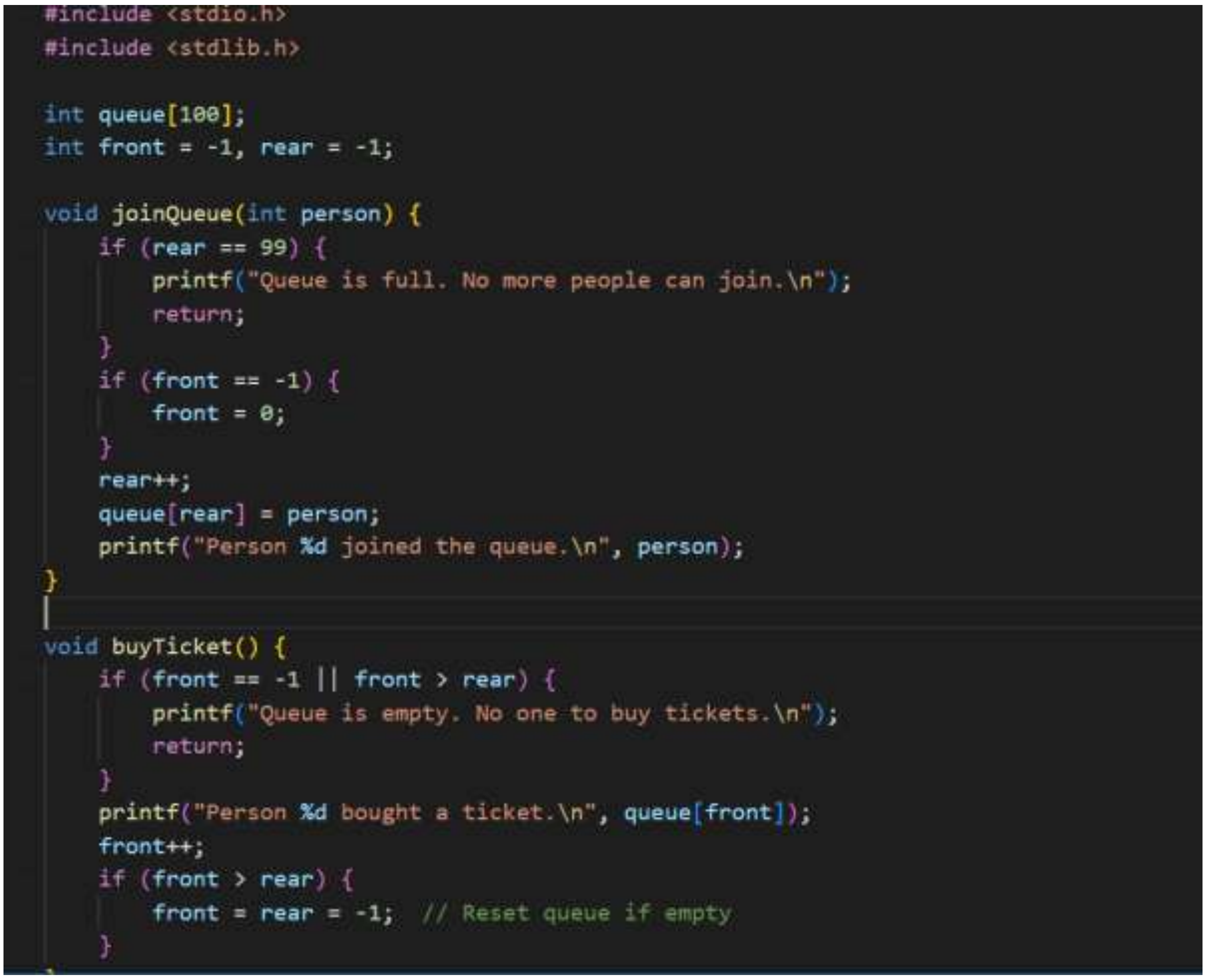
```
3. View Print Jobs
4. Exit
Enter your choice: 1
Enter job ID: 45
Enter job description (single word only): documnt
Print job 45 added: documnt

Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
3. View Print Jobs
4. Exit
Enter your choice: 3
Print jobs in queue:
Job ID: 23, Description: hr
Job ID: 45, Description: documnt

Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
3. View Print Jobs
4. Exit
Enter your choice: 2
Enter job ID to cancel: 45
Print job 45 canceled.

Print Job Scheduler
```



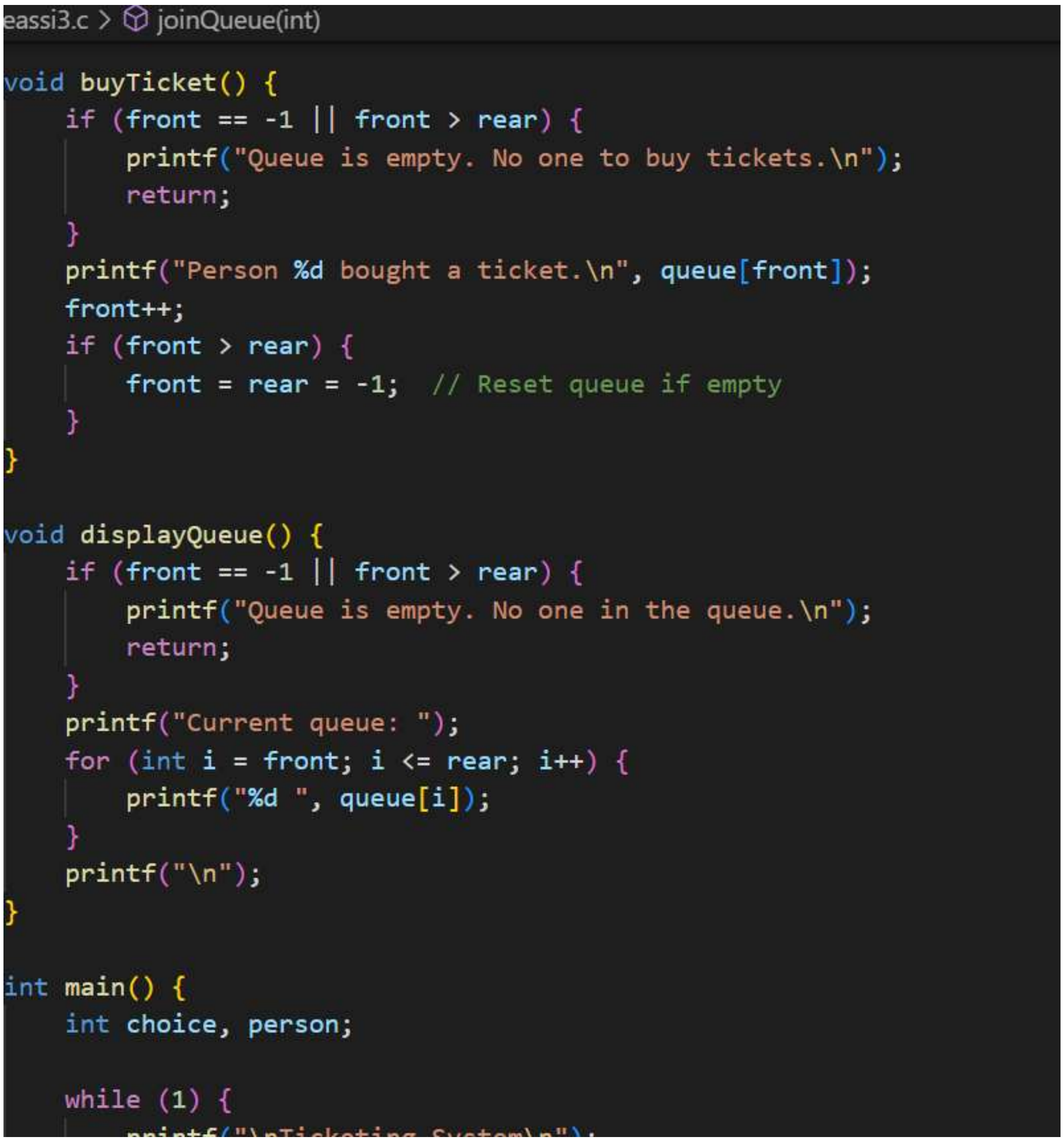
```
Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
3. View Print Jobs
4. Exit
Enter your choice: 3
Print jobs in queue:
Job ID: 23, Description: hr
Job ID: 45, Description: documnt

Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
3. View Print Jobs
4. Exit
Enter your choice: 2
Enter job ID to cancel: 45
Print job 45 canceled.

Print Job Scheduler
1. Add Print Job
2. Cancel Print Job
3. View Print Jobs
4. Exit
Enter your choice:
```

5. 3.Design a Ticketing System

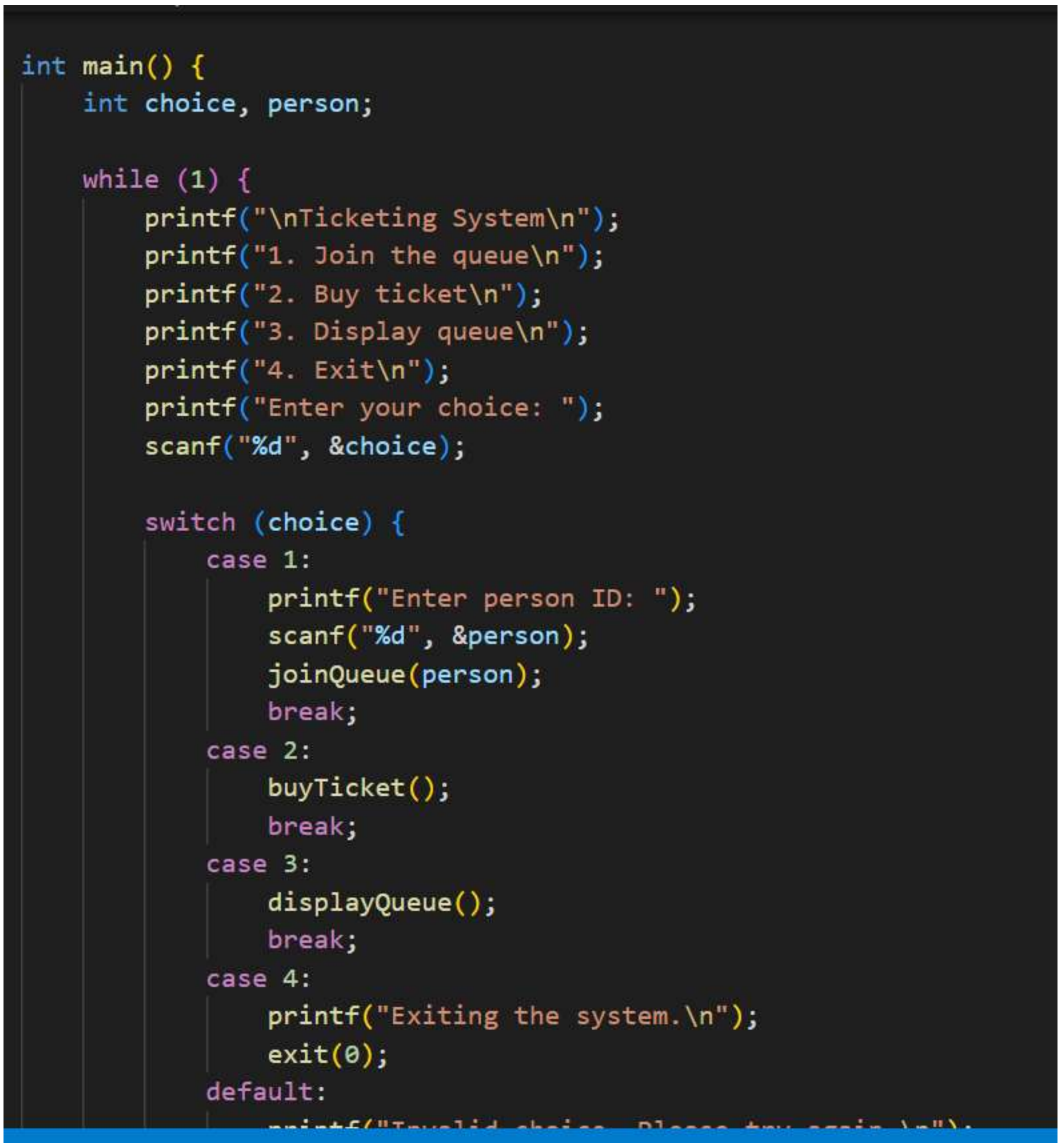
Simulate a ticketing system where people join a queue to buy tickets. Implement functionality for people to join the queue, buy tickets, and display the queue's current state.



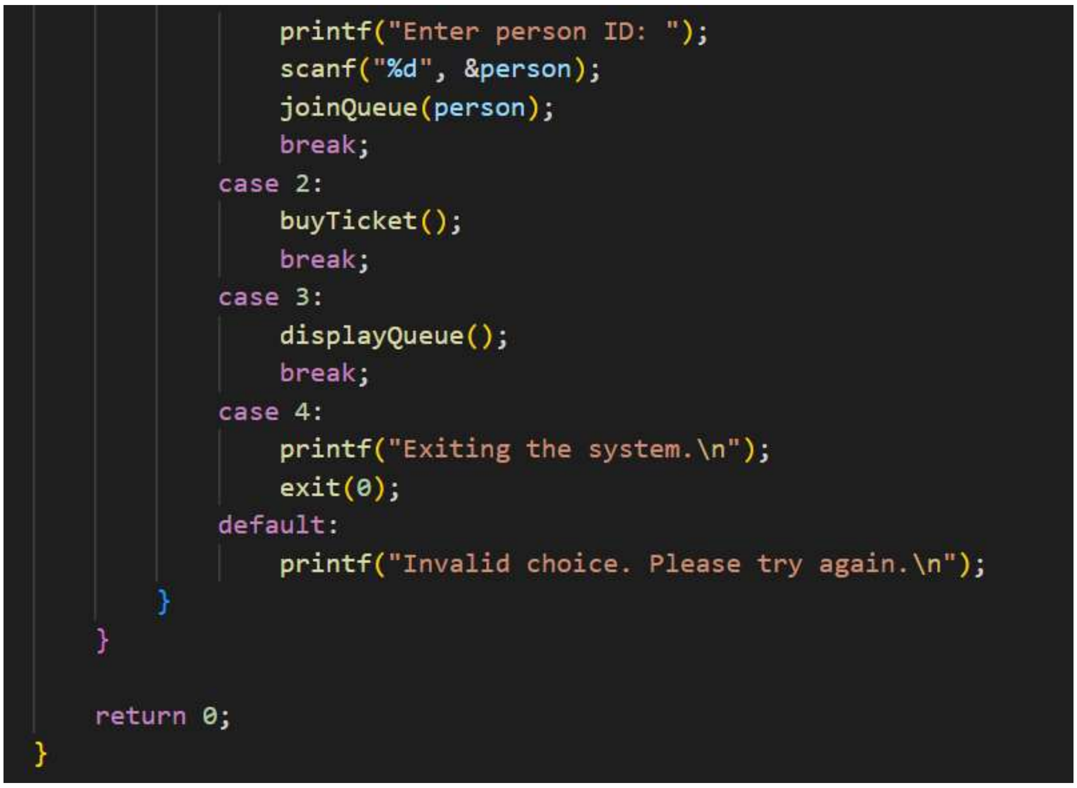
```c
#include <stdio.h>
#include <stdlib.h>

int queue[100];
int front = -1, rear = -1;

void joinQueue(int person) {
    if (rear == 99) {
        printf("Queue is full. No more people can join.\n");
        return;
    }
    if (front == -1) {
        front = 0;
    }
    rear++;
    queue[rear] = person;
    printf("Person %d joined the queue.\n", person);
}

void buyTicket() {
    if (front == -1 || front > rear) {
        printf("Queue is empty. No one to buy tickets.\n");
        return;
    }
    printf("Person %d bought a ticket.\n", queue[front]);
    front++;
    if (front > rear) {
        front = rear = -1;  // Reset queue if empty
    }
```


eassi3.c > joinQueue(int)

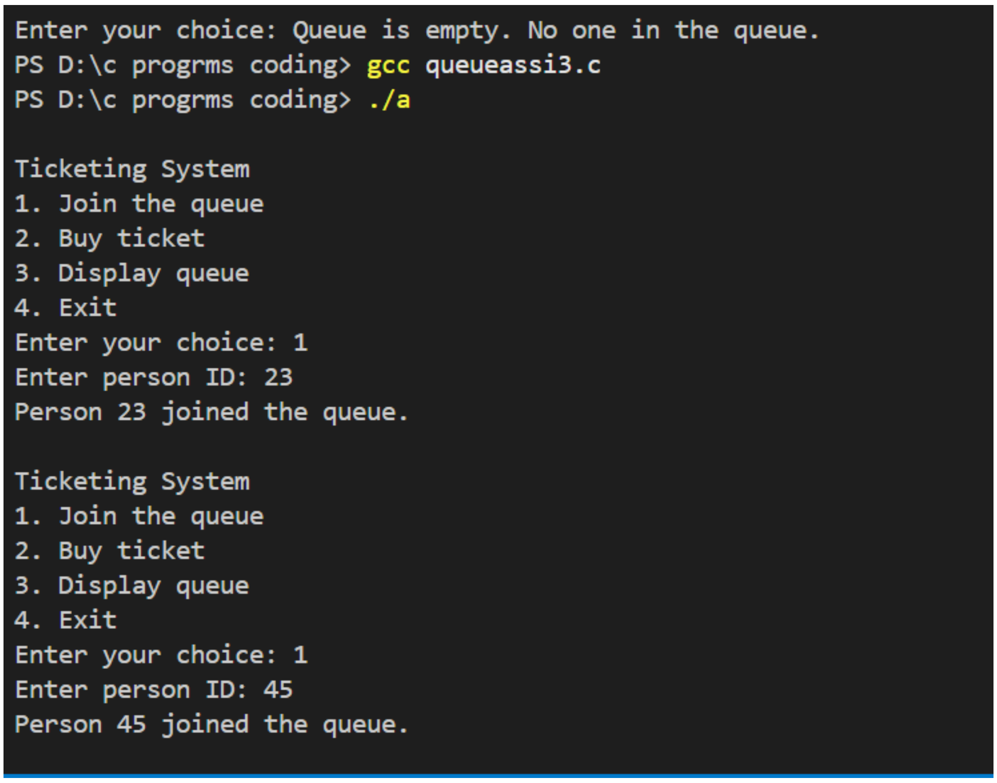
```c
void buyTicket() {
    if (front == -1 || front > rear) {
        printf("Queue is empty. No one to buy tickets.\n");
        return;
    }
    printf("Person %d bought a ticket.\n", queue[front]);
    front++;
    if (front > rear) {
        front = rear = -1;  // Reset queue if empty
    }
}

void displayQueue() {
    if (front == -1 || front > rear) {
        printf("Queue is empty. No one in the queue.\n");
        return;
    }
    printf("Current queue: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int choice, person;

    while (1) {
        printf("\nTicketing System\n");
```



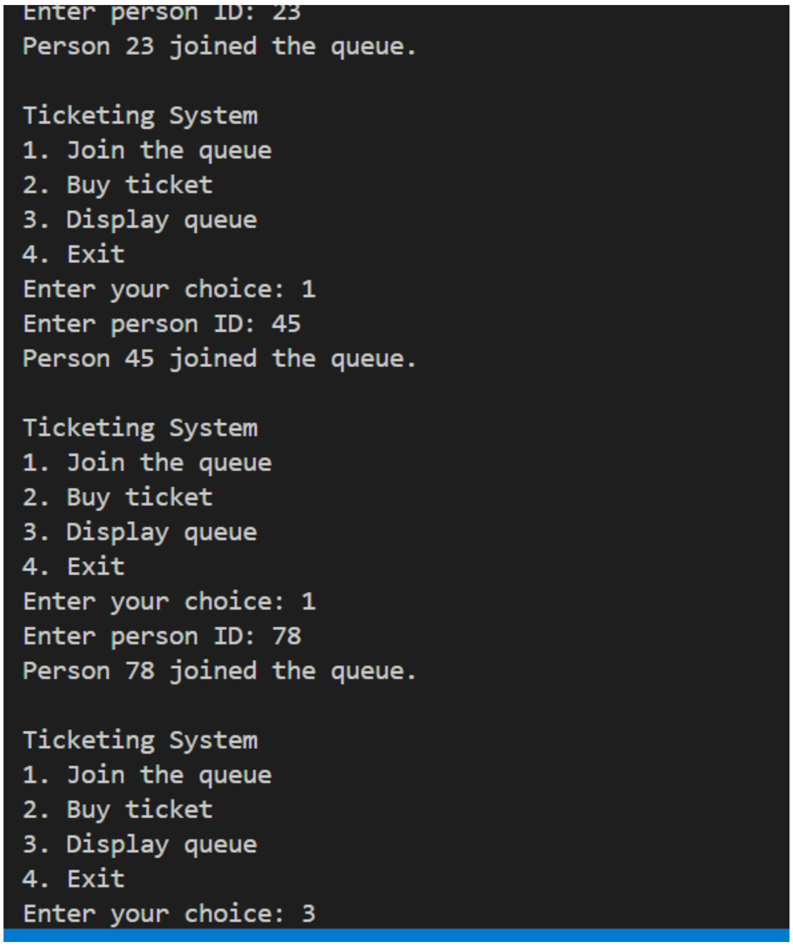
```c
int main() {
    int choice, person;

    while (1) {
        printf("\nTicketing System\n");
        printf("1. Join the queue\n");
        printf("2. Buy ticket\n");
        printf("3. Display queue\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter person ID: ");
                scanf("%d", &person);
                joinQueue(person);
                break;
            case 2:
                buyTicket();
                break;
            case 3:
                displayQueue();
                break;
            case 4:
                printf("Exiting the system.\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
```



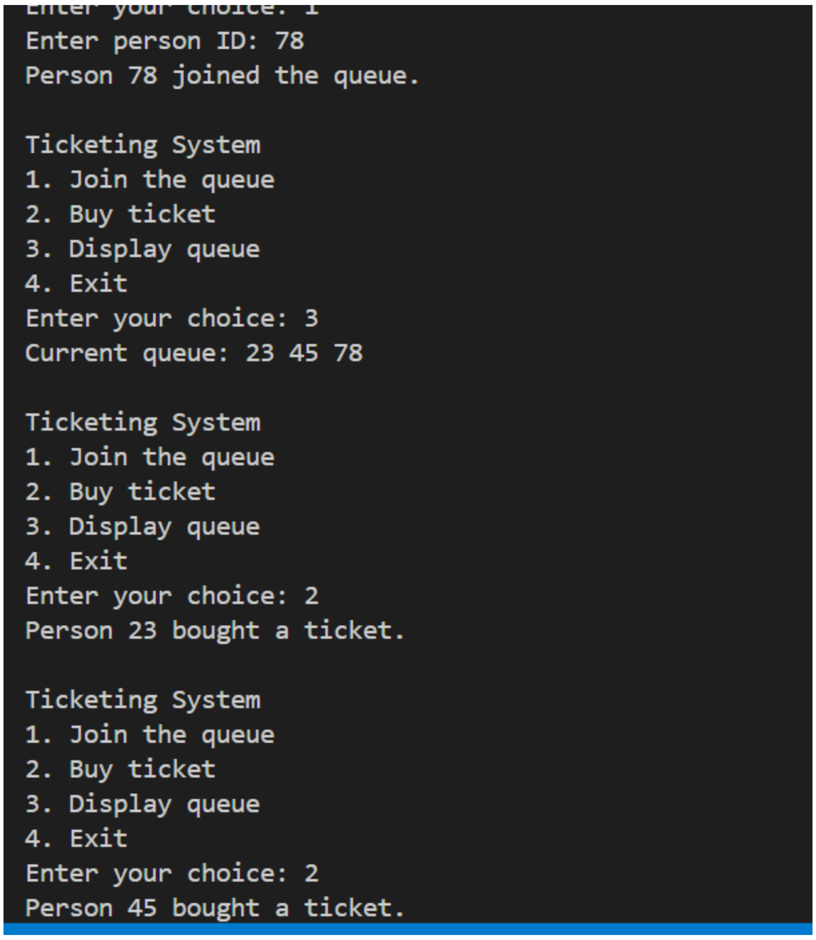
```c
                printf("Enter person ID: ");
                scanf("%d", &person);
                joinQueue(person);
                break;
            case 2:
                buyTicket();
                break;
            case 3:
                displayQueue();
                break;
            case 4:
                printf("Exiting the system.\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```



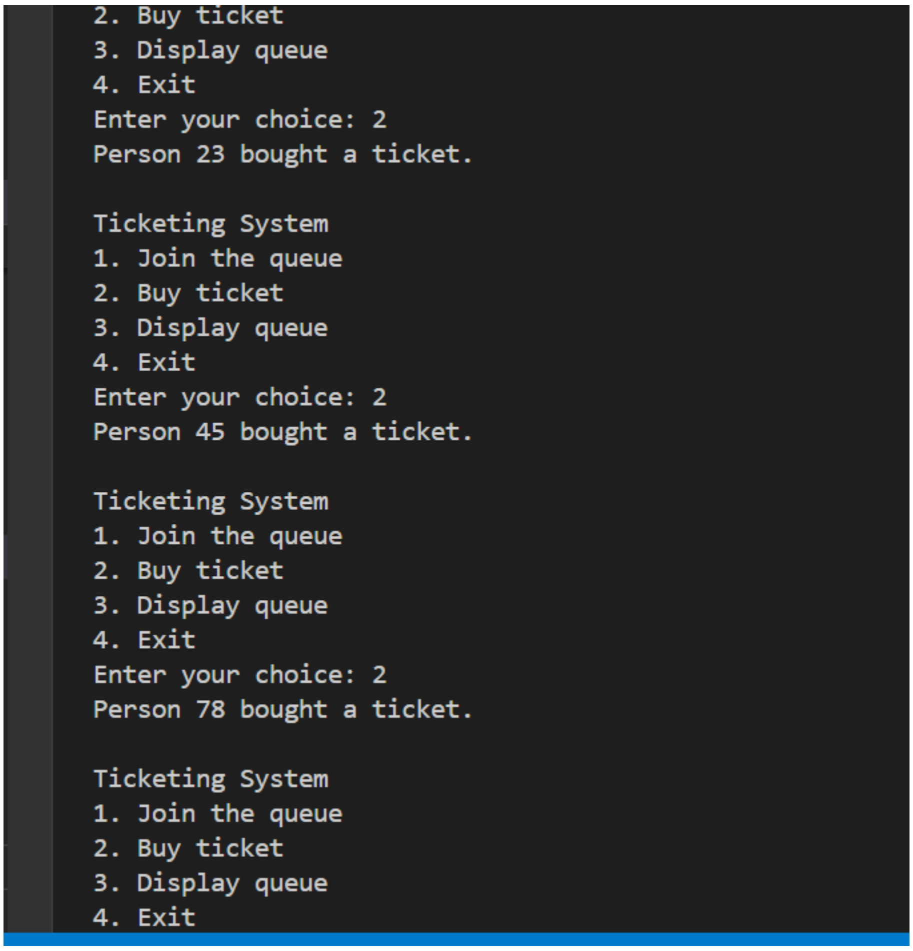
```
Enter your choice: Queue is empty. No one in the queue.
PS D:\c progrms coding> gcc queueassi3.c
PS D:\c progrms coding> ./a

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 1
Enter person ID: 23
Person 23 joined the queue.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 1
Enter person ID: 45
Person 45 joined the queue.
```



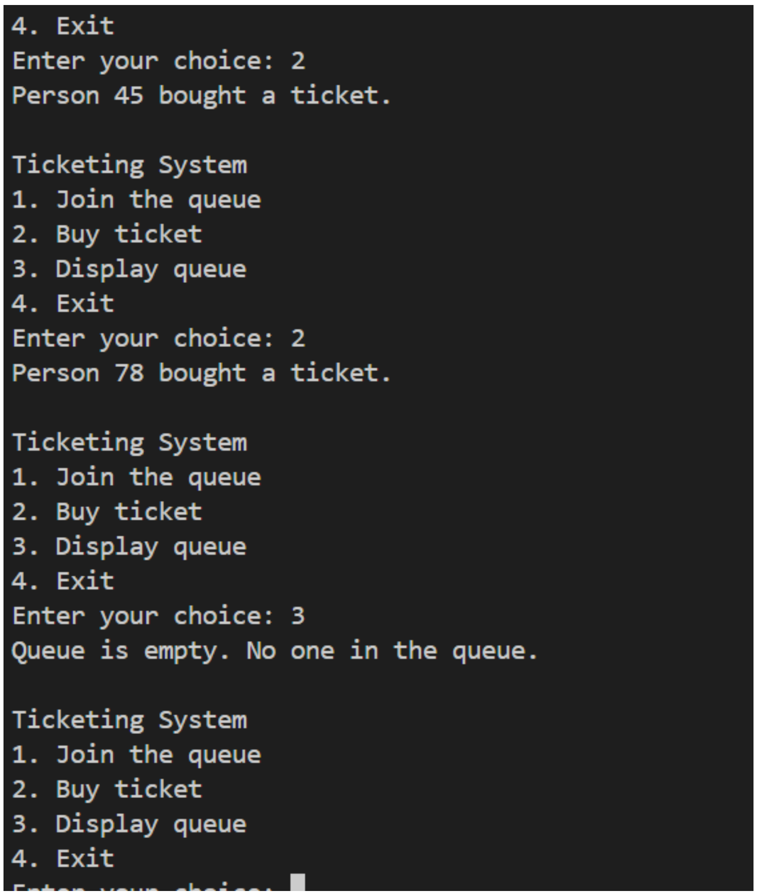
```
Enter person ID: 23
Person 23 joined the queue.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 1
Enter person ID: 45
Person 45 joined the queue.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 1
Enter person ID: 78
Person 78 joined the queue.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 3
```



```
Enter your choice: 1
Enter person ID: 78
Person 78 joined the queue.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 3
Current queue: 23 45 78

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 2
Person 23 bought a ticket.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 2
Person 45 bought a ticket.
```

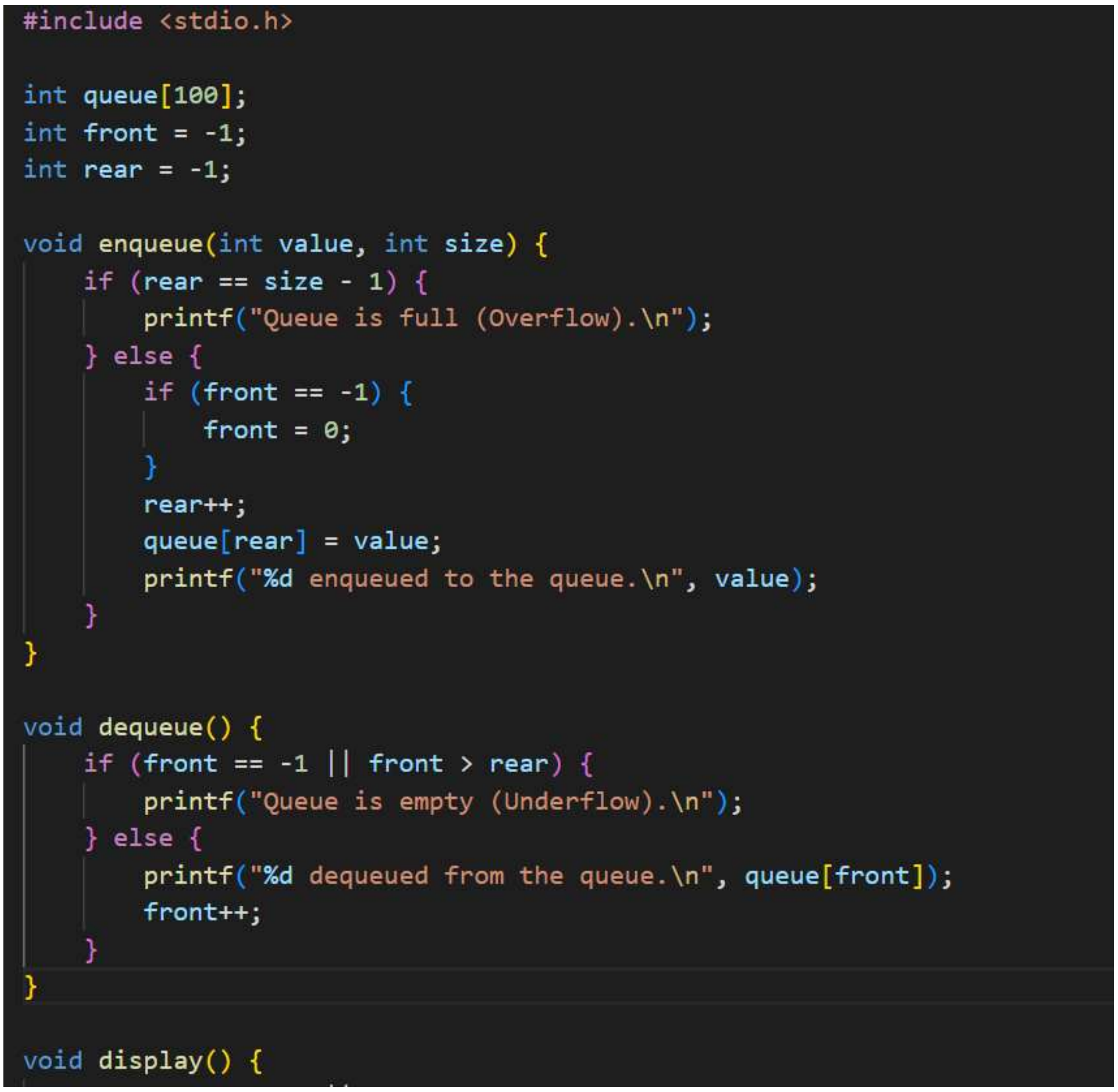


```
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 2
Person 23 bought a ticket.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 2
Person 45 bought a ticket.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 2
Person 78 bought a ticket.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
```

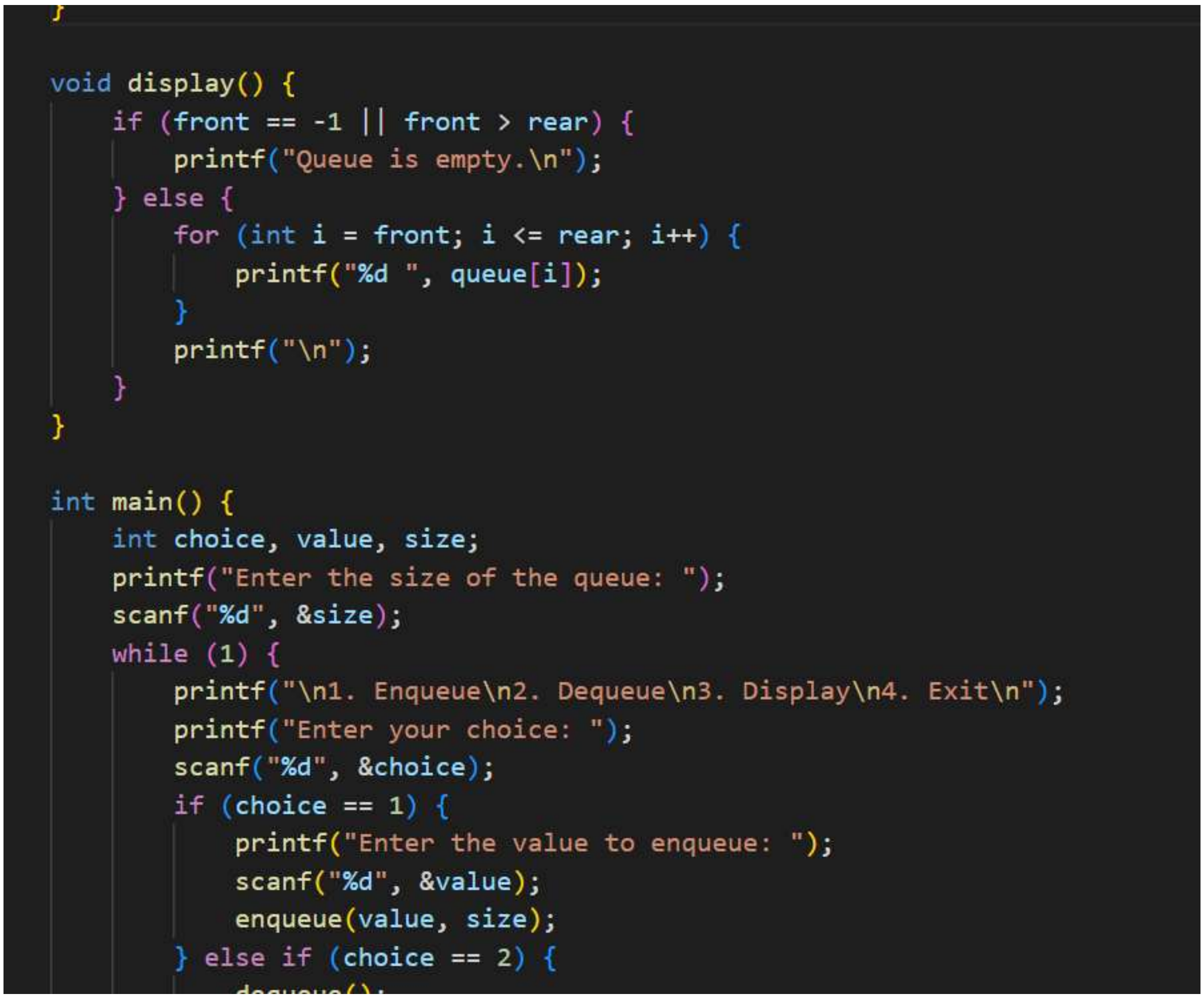


```
4. Exit
Enter your choice: 2
Person 45 bought a ticket.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 2
Person 78 bought a ticket.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
Enter your choice: 3
Queue is empty. No one in the queue.

Ticketing System
1. Join the queue
2. Buy ticket
3. Display queue
4. Exit
```

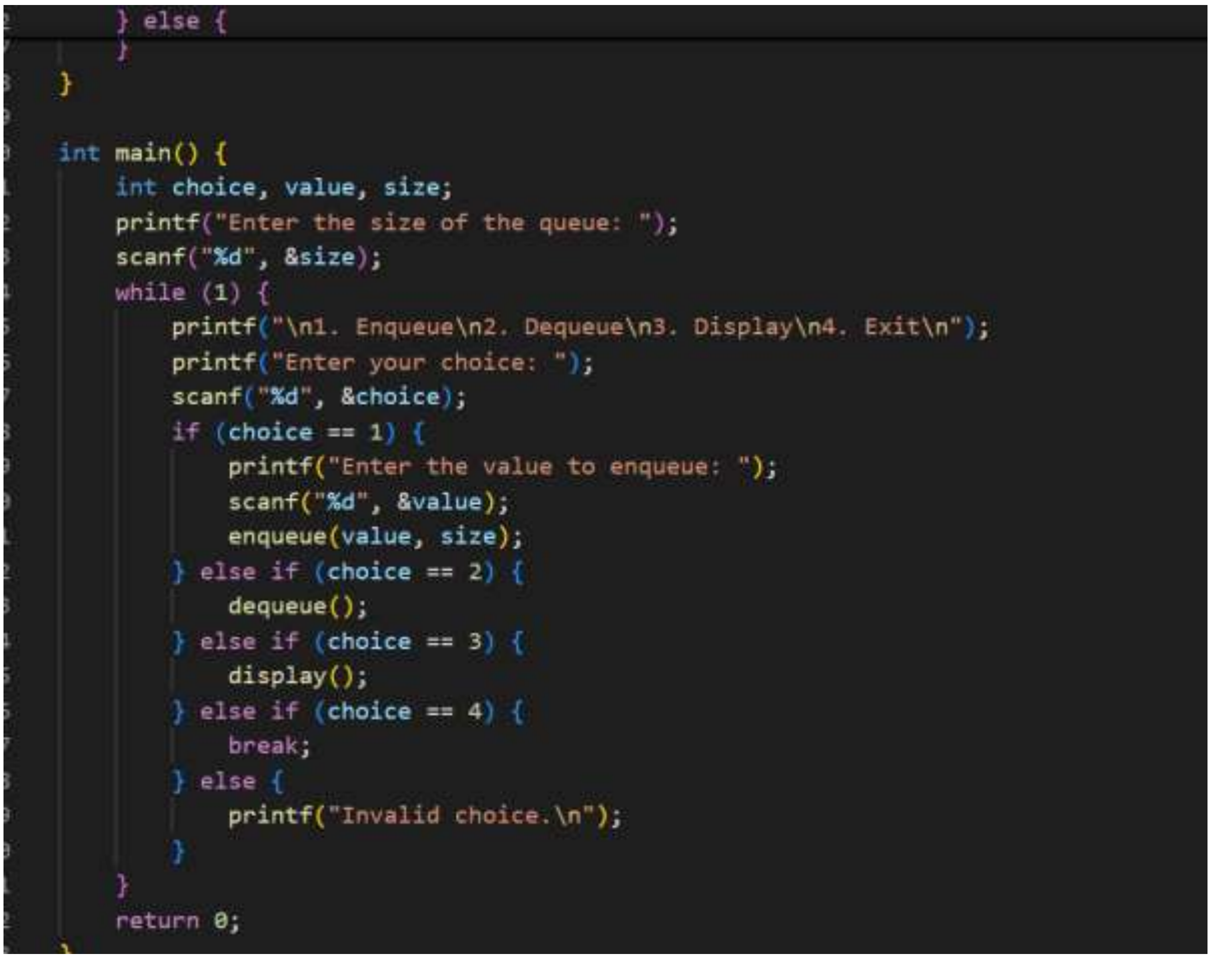
6.queue insertion deletion



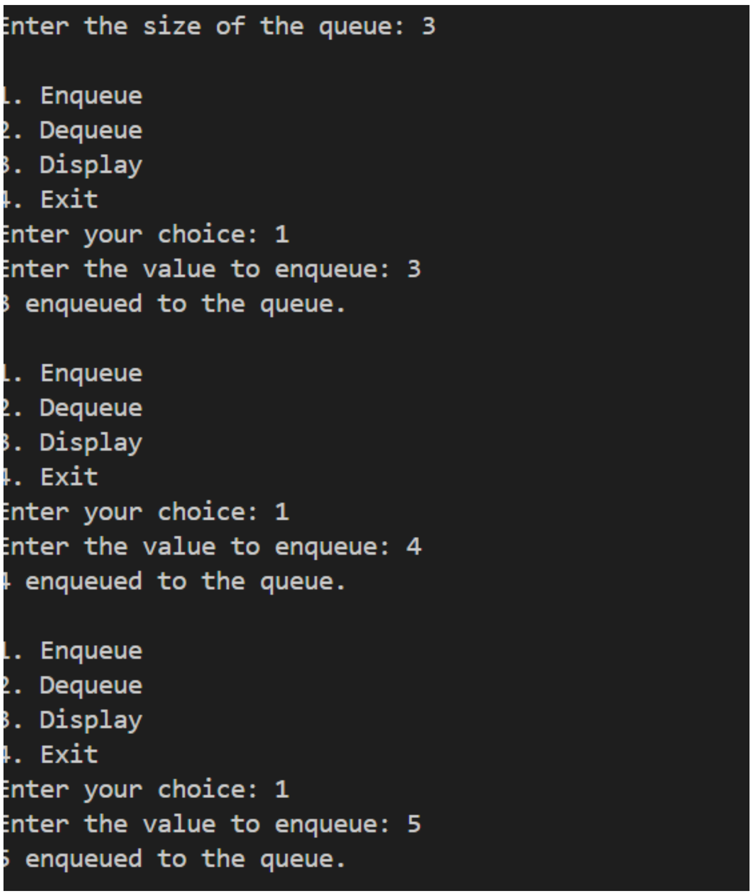
```c
#include <stdio.h>

int queue[100];
int front = -1;
int rear = -1;

void enqueue(int value, int size) {
    if (rear == size - 1) {
        printf("Queue is full (Overflow).\n");
    } else {
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("%d enqueued to the queue.\n", value);
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue is empty (Underflow).\n");
    } else {
        printf("%d dequeued from the queue.\n", queue[front]);
        front++;
    }
}

void display() {
```



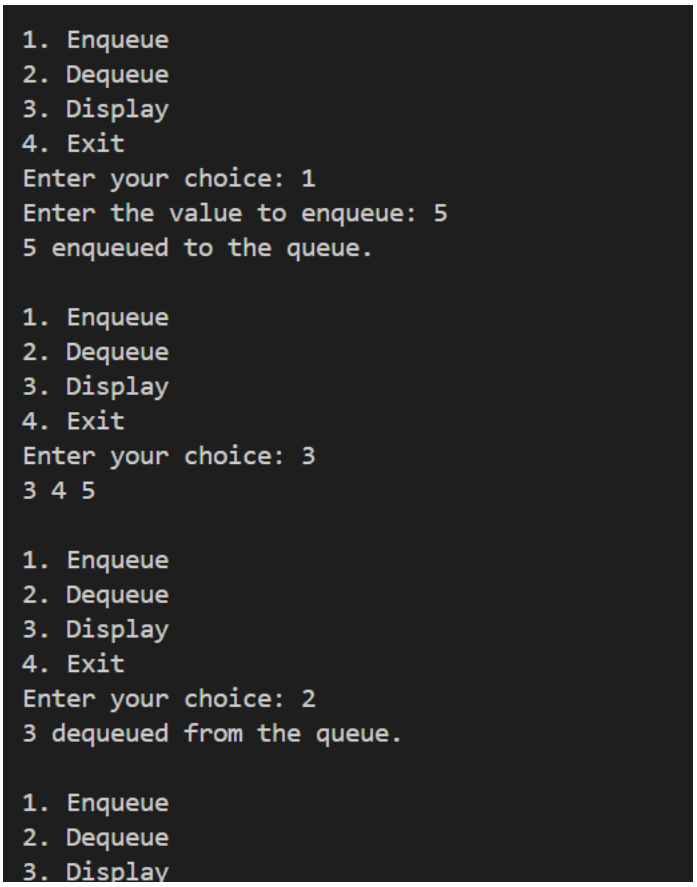
```
}

void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
    } else {
        for (int i = front; i <= rear; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice, value, size;
    printf("Enter the size of the queue: ");
    scanf("%d", &size);
    while (1) {
        printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        if (choice == 1) {
            printf("Enter the value to enqueue: ");
            scanf("%d", &value);
            enqueue(value, size);
        } else if (choice == 2) {
            dequeue();
```



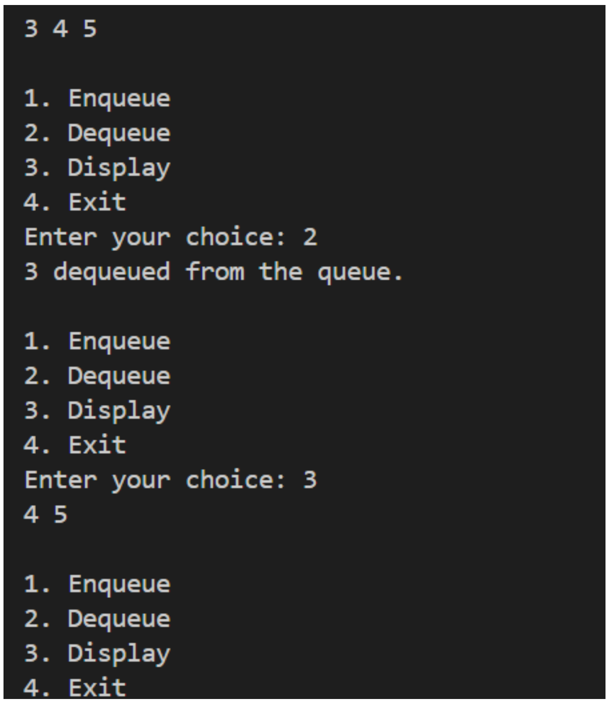
```c
        } else {
        }
    }

    int main() {
        int choice, value, size;
        printf("Enter the size of the queue: ");
        scanf("%d", &size);
        while (1) {
            printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n");
            printf("Enter your choice: ");
            scanf("%d", &choice);
            if (choice == 1) {
                printf("Enter the value to enqueue: ");
                scanf("%d", &value);
                enqueue(value, size);
            } else if (choice == 2) {
                dequeue();
            } else if (choice == 3) {
                display();
            } else if (choice == 4) {
                break;
            } else {
                printf("Invalid choice.\n");
            }
        }
        return 0;
```



```
nter the size of the queue: 3

. Enqueue
. Dequeue
. Display
. Exit
nter your choice: 1
nter the value to enqueue: 3
 enqueued to the queue.

. Enqueue
. Dequeue
. Display
. Exit
nter your choice: 1
nter the value to enqueue: 4
 enqueued to the queue.

. Enqueue
. Dequeue
. Display
. Exit
nter your choice: 1
nter the value to enqueue: 5
 enqueued to the queue.
```



```
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the value to enqueue: 5
5 enqueued to the queue.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
3 4 5

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
3 dequeued from the queue.

1. Enqueue
2. Dequeue
3. Display
```



```
3 4 5

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
3 dequeued from the queue.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
4 5

1. Enqueue
2. Dequeue
3. Display
4. Exit
```