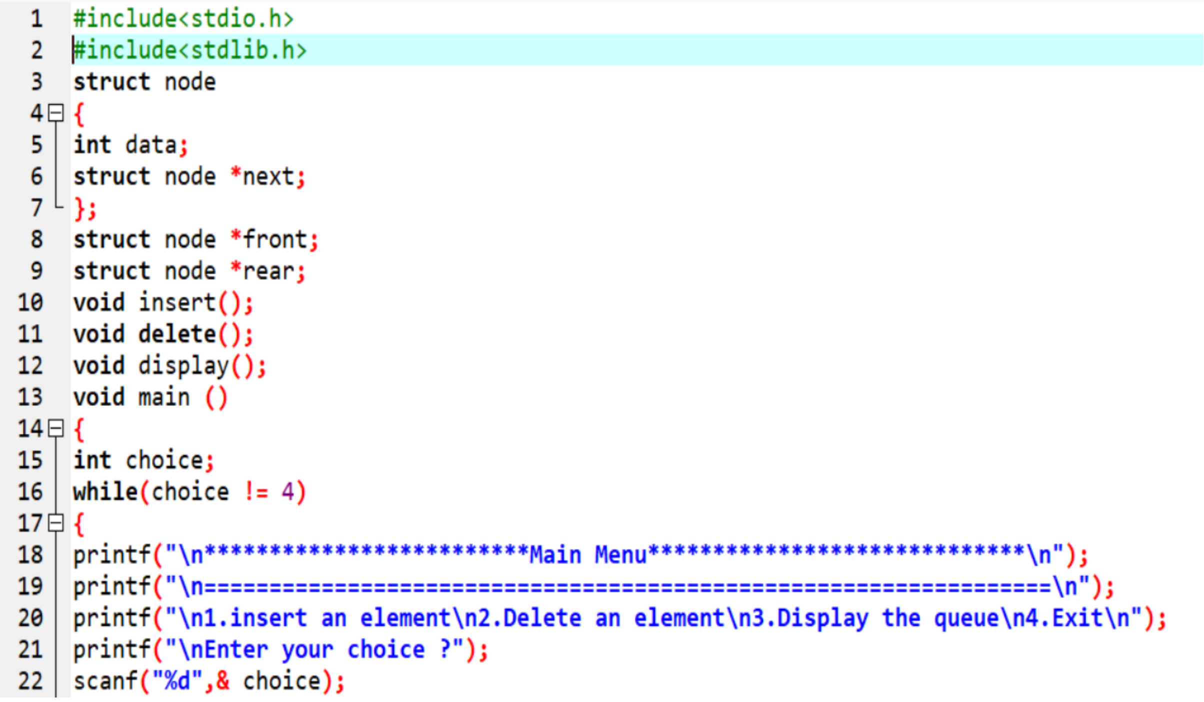
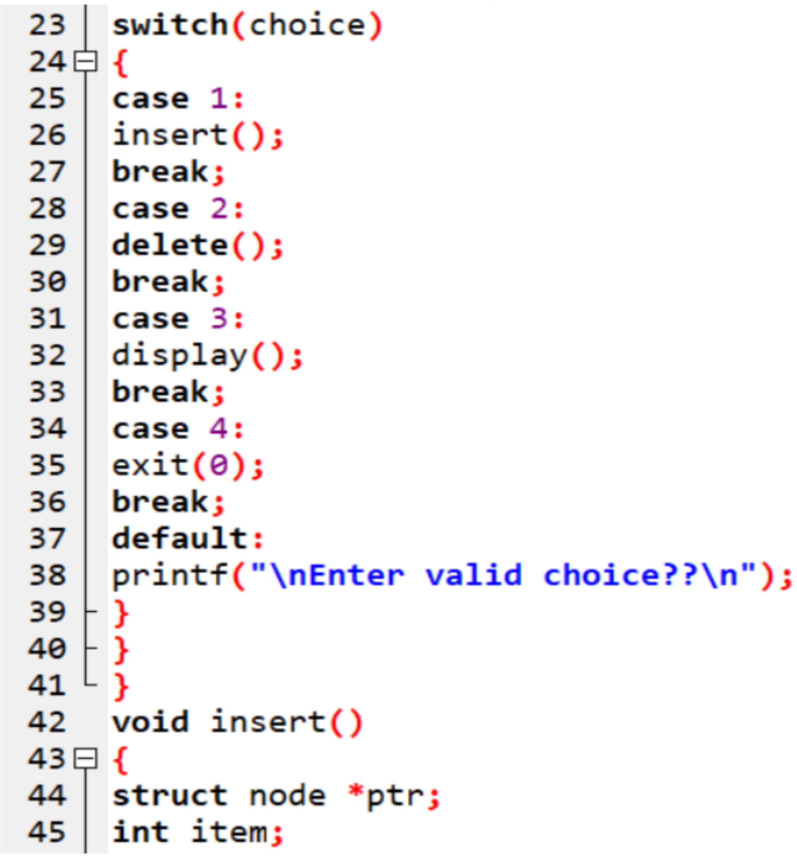
Experiment 3: Write C programs to implement a double ended queue ADT using

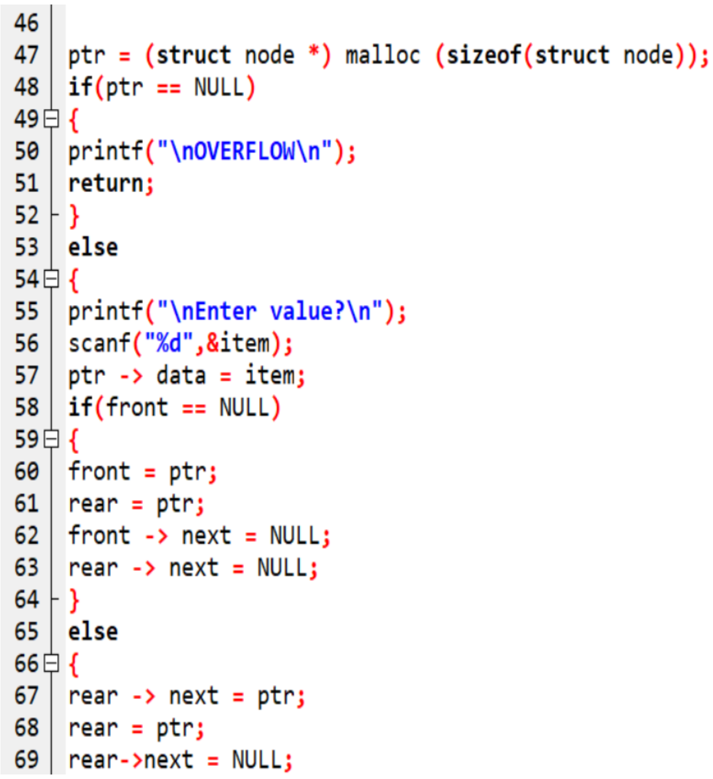
i) array



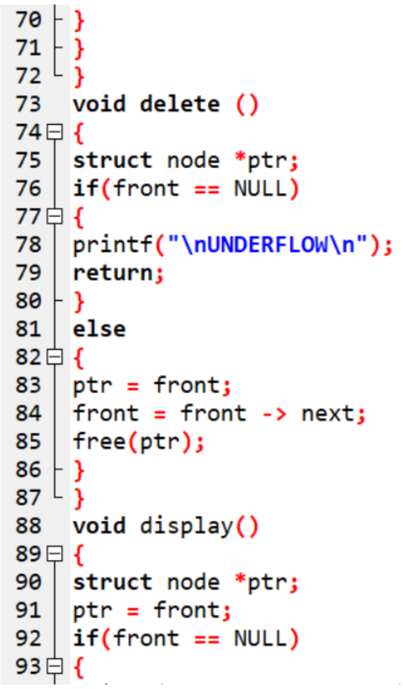
```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *front;
struct node *rear;
void insert();
void delete();
void display();
void main ()
{
int choice;
while(choice != 4)
{
printf("\n*************************Main Menu*****************************\n");
printf("\n=================================================================\n");
printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");
printf("\nEnter your choice ?");
scanf("%d",& choice);
```



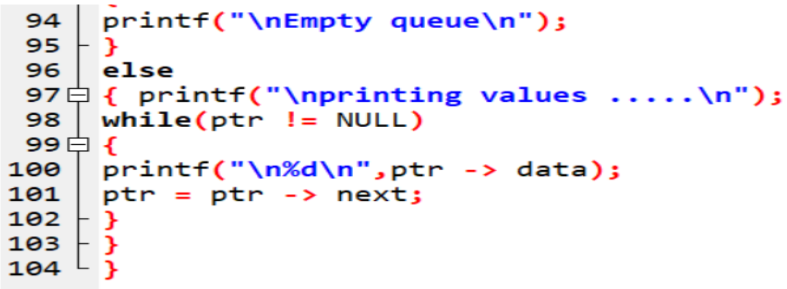
```c
switch(choice)
{
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
exit(0);
break;
default:
printf("\nEnter valid choice??\n");
}
}
}
void insert()
{
struct node *ptr;
int item;
```



```
ptr = (struct node *) malloc (sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW\n");
return;
}
else
{
printf("\nEnter value?\n");
scanf("%d",&item);
ptr -> data = item;
if(front == NULL)
{
front = ptr;
rear = ptr;
front -> next = NULL;
rear -> next = NULL;
}
else
{
rear -> next = ptr;
rear = ptr;
rear->next = NULL;
```

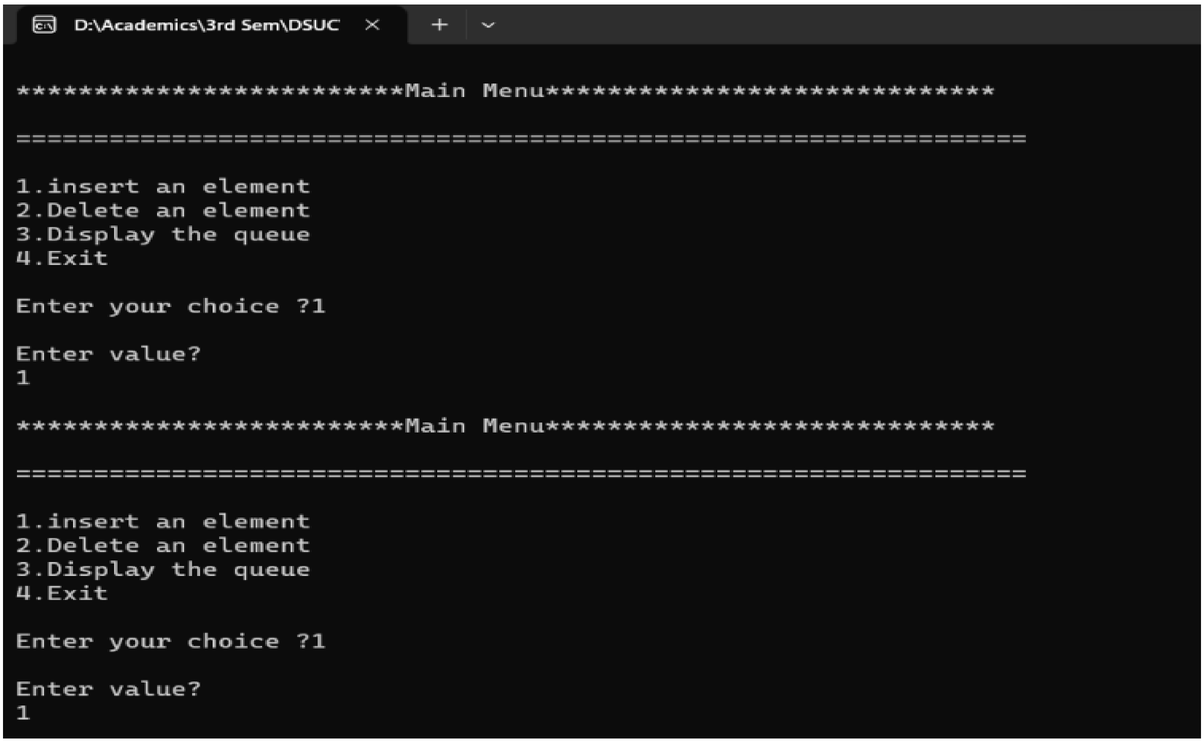


```
}
}
}
void delete ()
{
struct node *ptr;
if(front == NULL)
{
printf("\nUNDERFLOW\n");
return;
}
else
{
ptr = front;
front = front -> next;
free(ptr);
}
}
void display()
{
struct node *ptr;
ptr = front;
if(front == NULL)
{
```

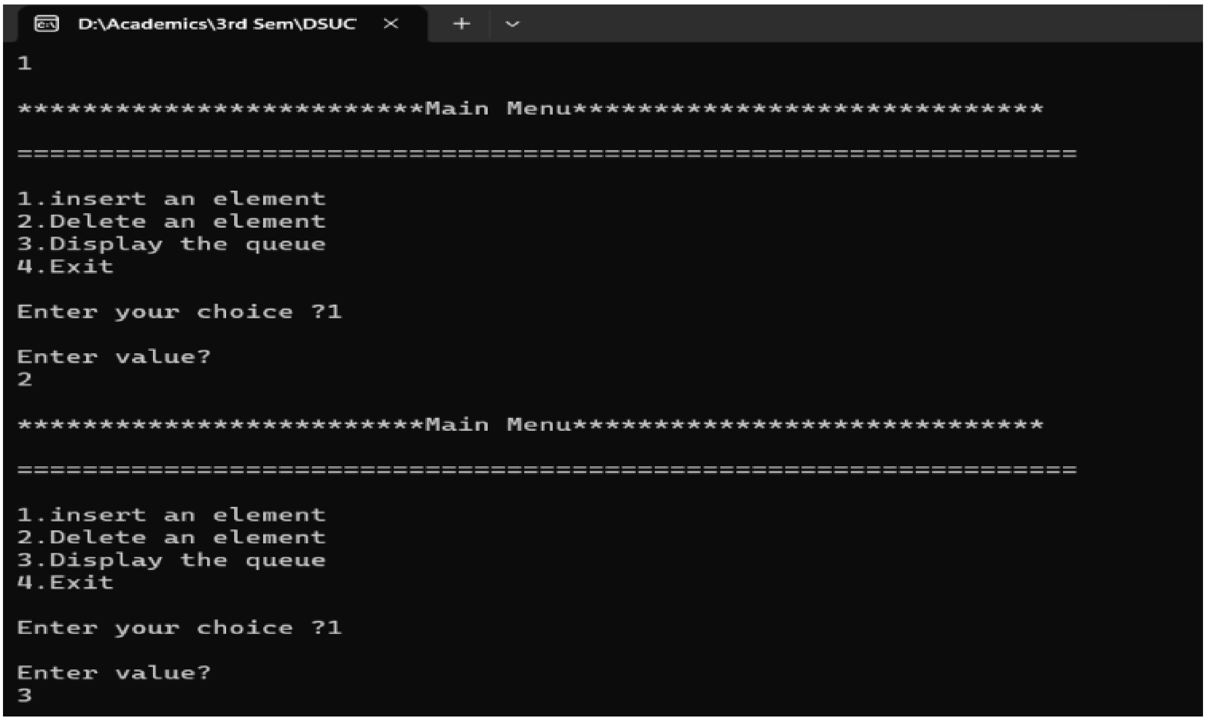


```
printf("\nEmpty queue\n");
}
else
{ printf("\nprinting values .....\n");
while(ptr != NULL)
{
printf("\n%d\n",ptr -> data);
ptr = ptr -> next;
}
}
}
```

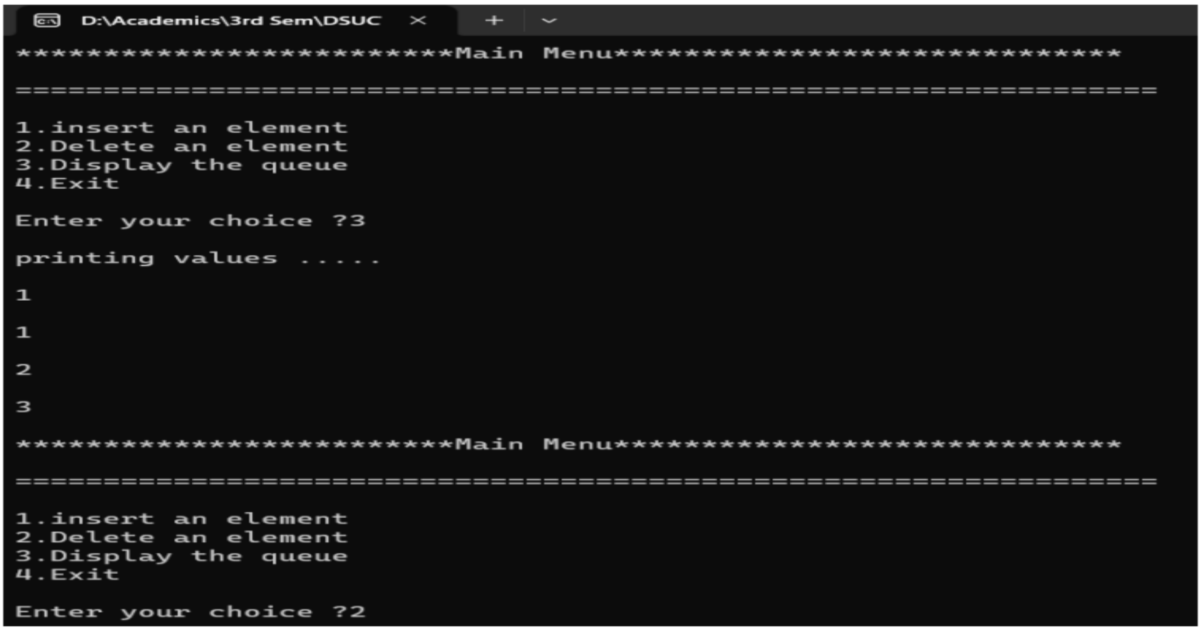
Output:



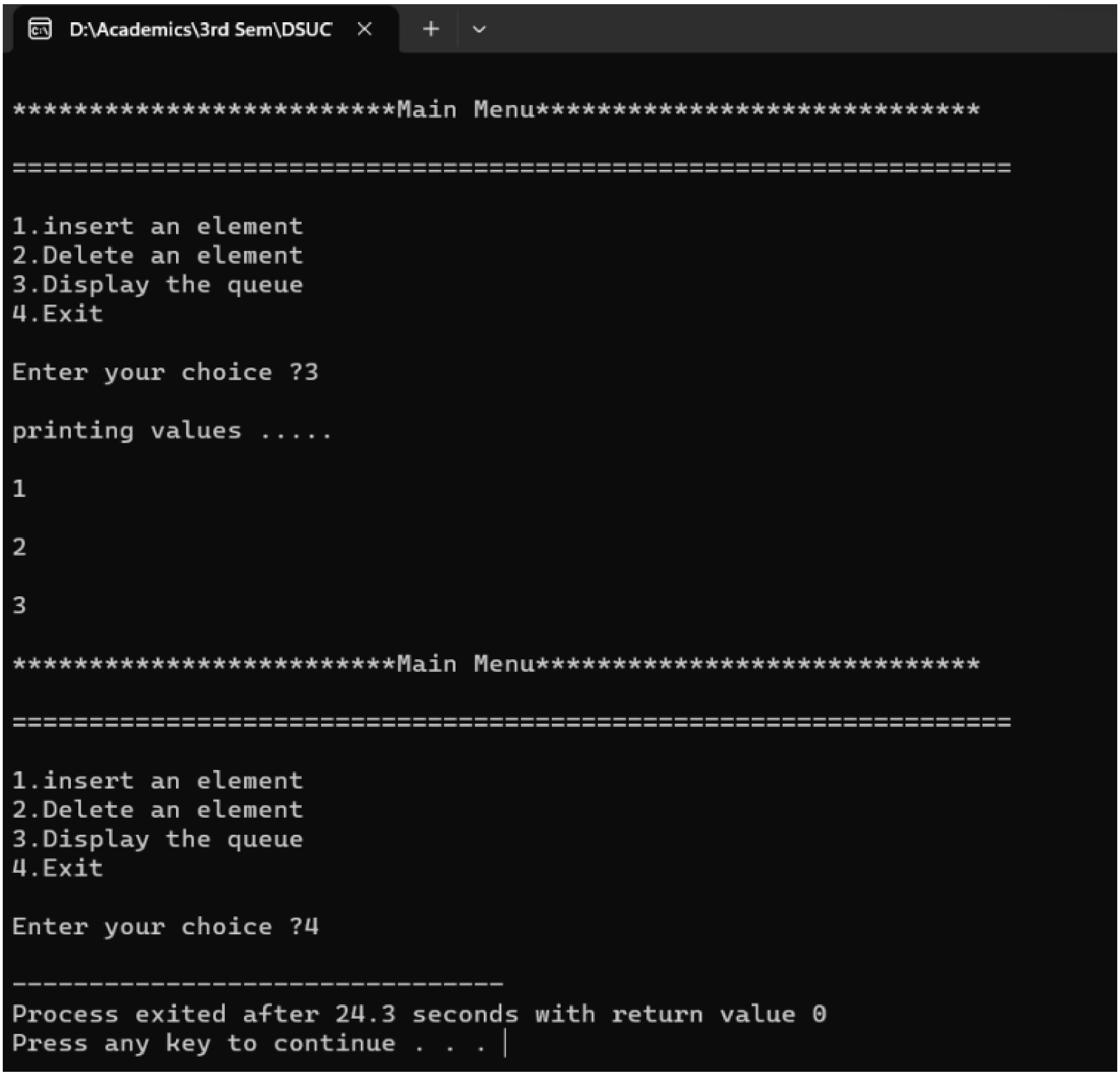
```
*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?1

Enter value?
1

*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?1

Enter value?
1
```



```
1

*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?1

Enter value?
2

*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?1

Enter value?
3
```

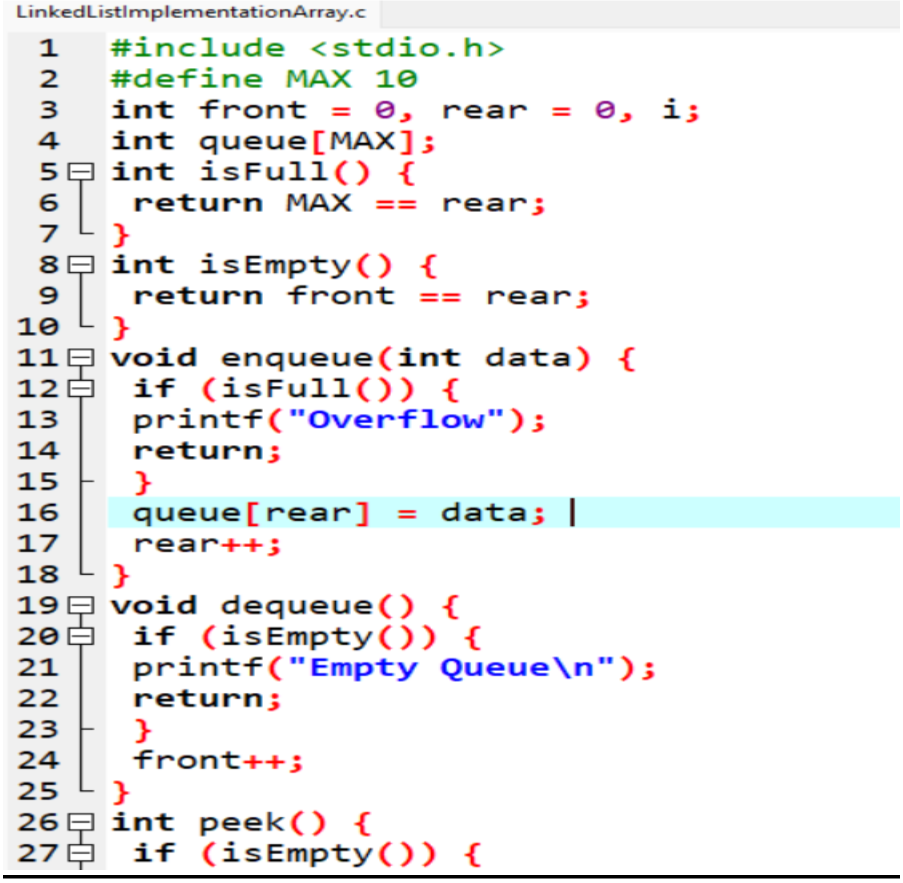


```
*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?3

printing values .....

1

1

2

3

*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?2
```

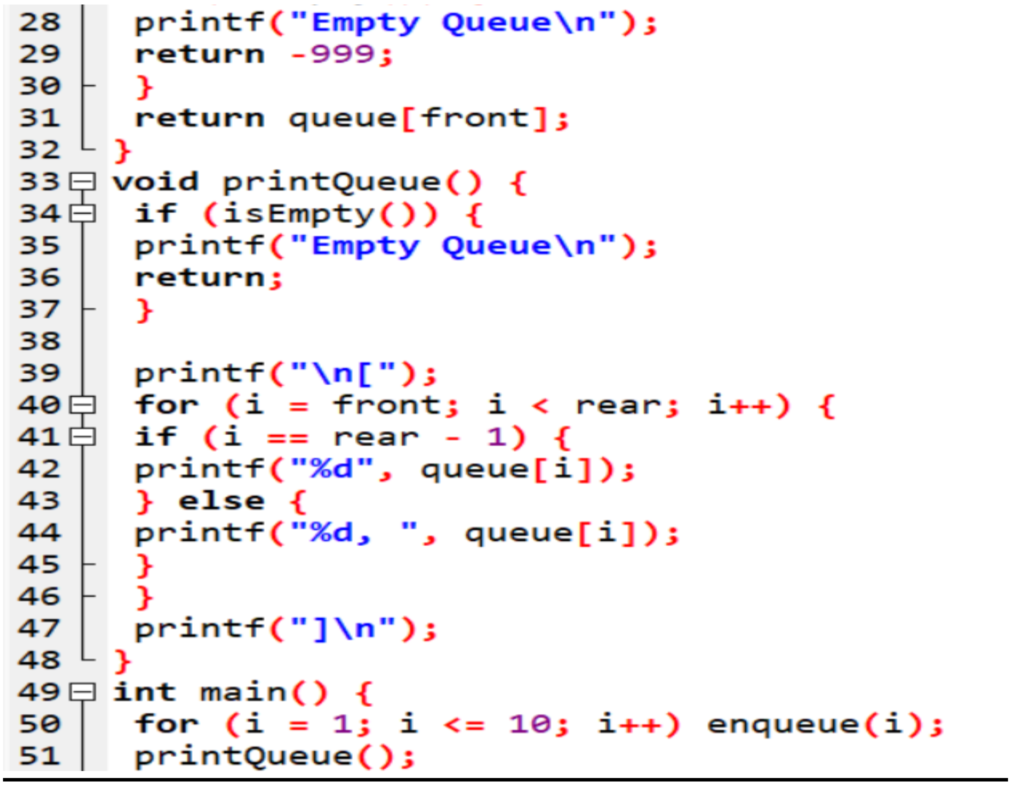


```
*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?3

printing values .....

1

2

3

*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?4

--------------------------------
Process exited after 24.3 seconds with return value 0
Press any key to continue . . .
```

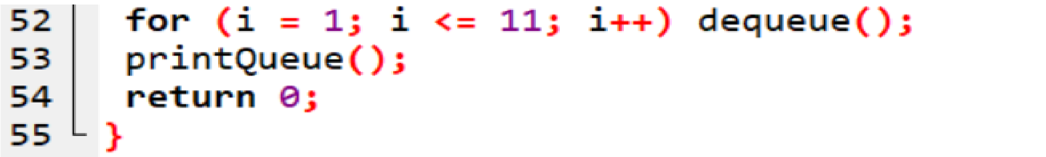
ii) doubly linked list respectively.



```c
#include <stdio.h>
#define MAX 10
int front = 0, rear = 0, i;
int queue[MAX];
int isFull() {
 return MAX == rear;
}
int isEmpty() {
 return front == rear;
}
void enqueue(int data) {
 if (isFull()) {
 printf("Overflow");
 return;
 }
 queue[rear] = data;
 rear++;
}
void dequeue() {
 if (isEmpty()) {
 printf("Empty Queue\n");
 return;
 }
 front++;
}
int peek() {
 if (isEmpty()) {
```

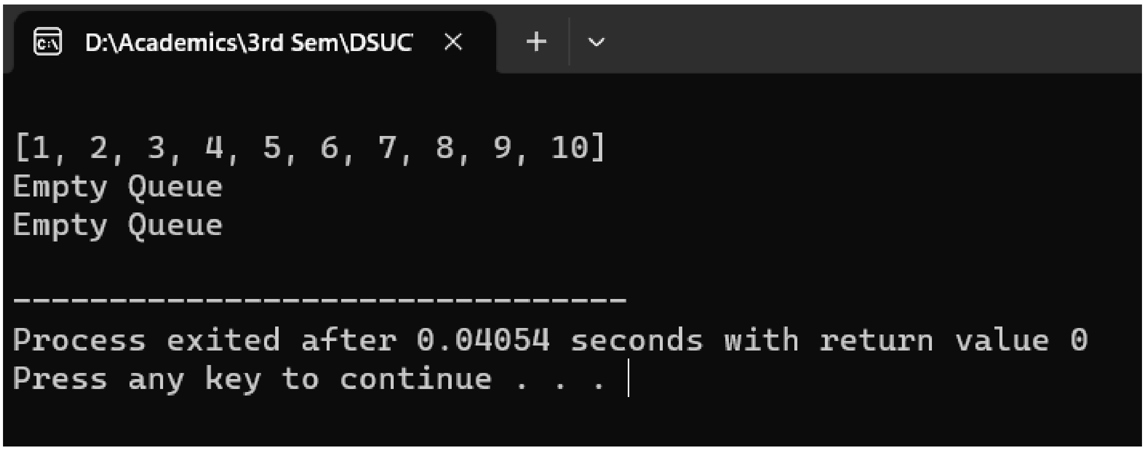


```c
 printf("Empty Queue\n");
 return -999;
 }
 return queue[front];
}
void printQueue() {
 if (isEmpty()) {
 printf("Empty Queue\n");
 return;
 }

 printf("\n[");
 for (i = front; i < rear; i++) {
 if (i == rear - 1) {
 printf("%d", queue[i]);
 } else {
 printf("%d, ", queue[i]);
 }
 }
 printf("]\n");
}
int main() {
 for (i = 1; i <= 10; i++) enqueue(i);
 printQueue();
```



```
    for (i = 1; i <= 11; i++) dequeue();
    printQueue();
    return 0;
}
```

Output:



```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Empty Queue
Empty Queue

--------------------------------
Process exited after 0.04054 seconds with return value 0
Press any key to continue . . .
```