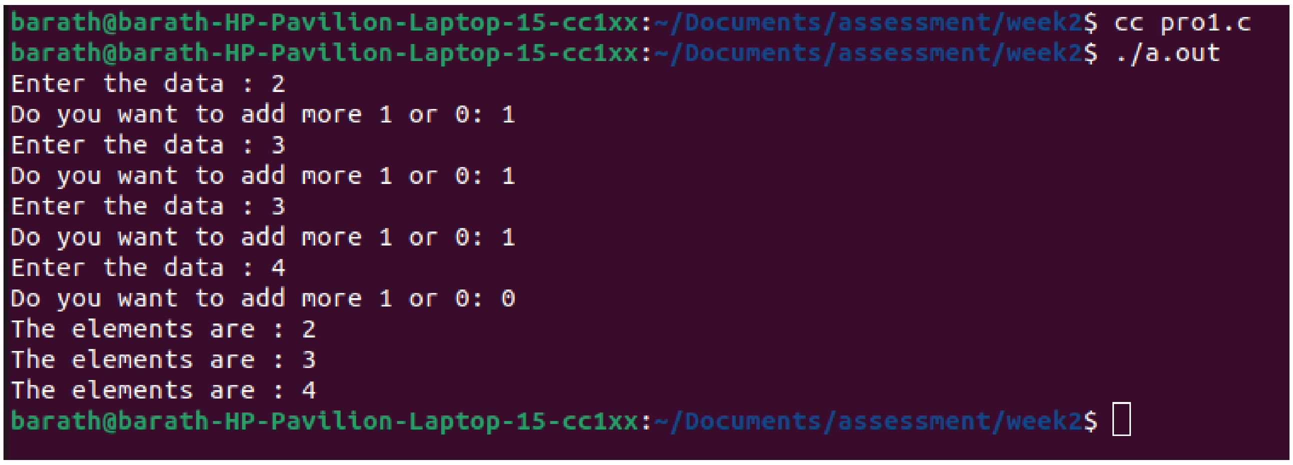
1. Remove Duplicates on Linked List

// Remove Duplicates from linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node* next;
};
struct node *newnode, *head;
void createLL()
{
        struct node *temp;
        int ch=1;
        while(ch){
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("Enter the data : ");
        scanf("%d", &newnode->data);
        newnode->next=0;
        if (head==NULL){
                head=newnode;
                temp=newnode;
        }
        else{
                temp->next=newnode;
                temp=newnode;
        }
        printf("Do you want to add more 1 or 0: ");
        scanf("%d", &ch);
        }
}
void removeduplicate(){
        struct node* current = head;
        struct node* temp;
     if (current == NULL)
     {
         printf("Empty List");
     }
     else
     {
         while (current->next != NULL)
         {
           if (current->data == current->next->data)
           {
                        temp = current->next->next;
                        free(current->next);
                        current->next = temp;
              }
           else
           {
```

```
                            current = current->next;
                 }
                  }
               }
}
void display()
{
            struct node* temp;
            temp=head;
            if(temp == NULL)
            {
             printf("Empty List");
        }
        else
        {
            while(temp != NULL)
            {
                        printf("The elements are : %d\n", temp->data);
                        temp=temp->next;
            }
        }
}
int main(){
            createLL();
            removeduplicate();
            display();
}
```

RESULT :



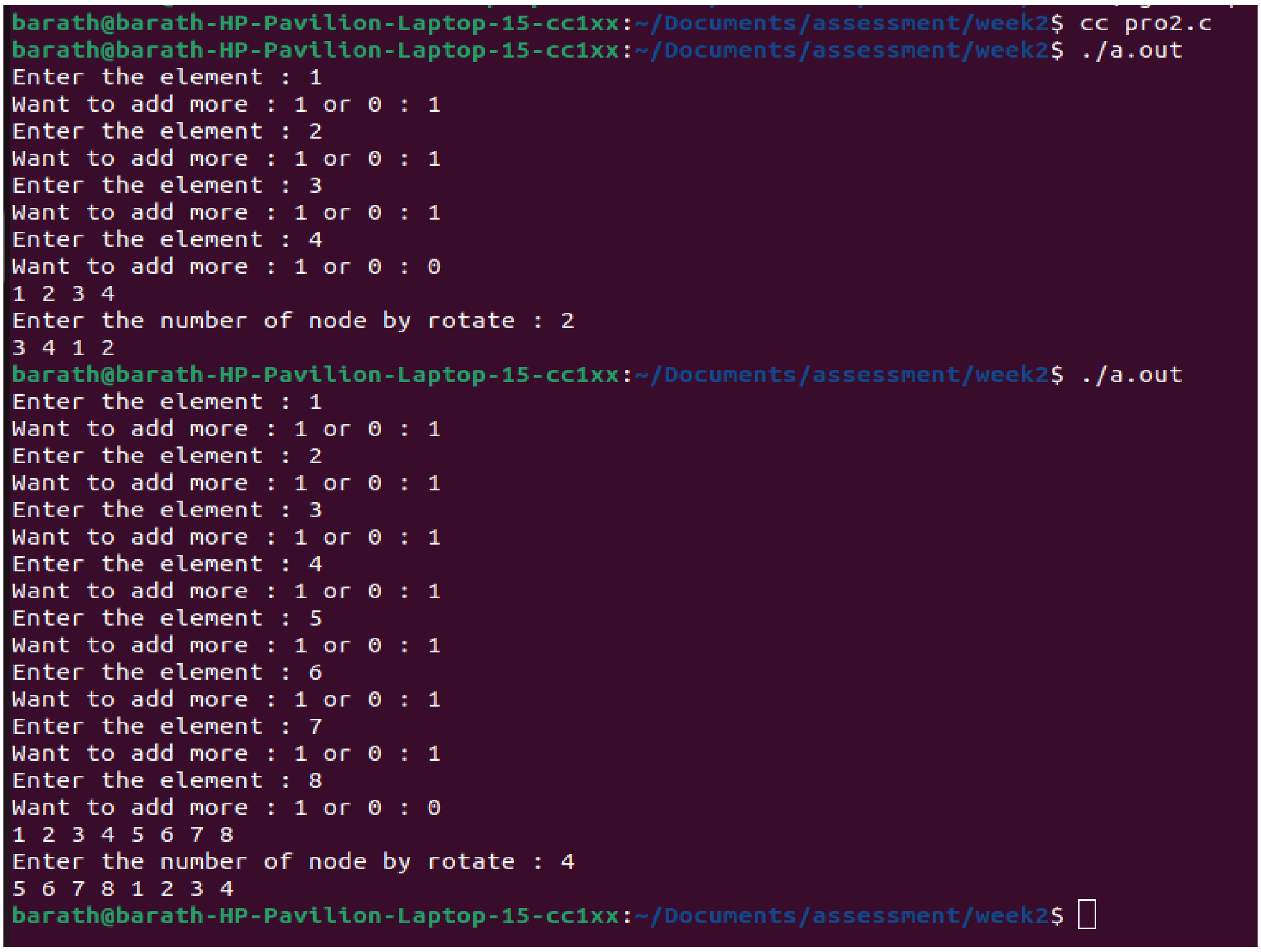
```
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ cc pro1.c
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ ./a.out
Enter the data : 2
Do you want to add more 1 or 0: 1
Enter the data : 3
Do you want to add more 1 or 0: 1
Enter the data : 3
Do you want to add more 1 or 0: 1
Enter the data : 4
Do you want to add more 1 or 0: 0
The elements are : 2
The elements are : 3
The elements are : 4
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ 
```

2.Rotate Doubly Linked List by N nodes

//Rotate Doubly Linked List by N nodes

```
#include<stdio.h>
#include<stdlib.h>
struct node{
        char data;
        struct node *prev;
        struct node *next;
};
struct node *newnode, *head;
void createDLL(){
        struct node *temp;
        int ch=1;
        int a;
        while(ch){
        printf("Enter the element : ");
        scanf("%d", &a);
        newnode=(struct node*)malloc(sizeof(struct node));
        newnode->data=a;
        newnode->prev=0;
        newnode->next=0;
        if (head == NULL)
        {
                head=newnode;
                temp=newnode;
        }
        else{
                temp->next=newnode;
                newnode->prev=temp;
                temp=newnode;
        }
        printf("Want to add more : 1 or 0 : ");
        scanf("%d", &ch);
        }
}
void rotate(){
        struct node *temp=head;
        int n;
        printf("Enter the number of node by rotate : ");
        scanf("%d", &n);
        while(temp->next != NULL)
        {
        temp=temp->next;
        }
        temp->next=head;
        head->prev=temp;
        int count=1;
        while(count <= n)
        {
        head=head->next;
        temp=temp->next;
        count++;
        }
        temp->next=NULL;
        head->prev=NULL;
}
void display(){
        struct node* temp=head;
        if(head == NULL){
                printf("List empty ");
        }
        else{
                while(temp->next != NULL){
                        printf("%d", temp->data);
                        temp=temp->next;
                }
                printf("%d", temp->data);
        }
}
int main(){
        createDLL();
        display();
        rotate();
        display();
}
```

RESULT :



```
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ cc pro2.c
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ ./a.out
Enter the element : 1
Want to add more : 1 or 0 : 1
Enter the element : 2
Want to add more : 1 or 0 : 1
Enter the element : 3
Want to add more : 1 or 0 : 1
Enter the element : 4
Want to add more : 1 or 0 : 0
1 2 3 4
Enter the number of node by rotate : 2
3 4 1 2
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ ./a.out
Enter the element : 1
Want to add more : 1 or 0 : 1
Enter the element : 2
Want to add more : 1 or 0 : 1
Enter the element : 3
Want to add more : 1 or 0 : 1
Enter the element : 4
Want to add more : 1 or 0 : 1
Enter the element : 5
Want to add more : 1 or 0 : 1
Enter the element : 6
Want to add more : 1 or 0 : 1
Enter the element : 7
Want to add more : 1 or 0 : 1
Enter the element : 8
Want to add more : 1 or 0 : 0
1 2 3 4 5 6 7 8
Enter the number of node by rotate : 4
5 6 7 8 1 2 3 4
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$
```

5.Reverse the elements using stack

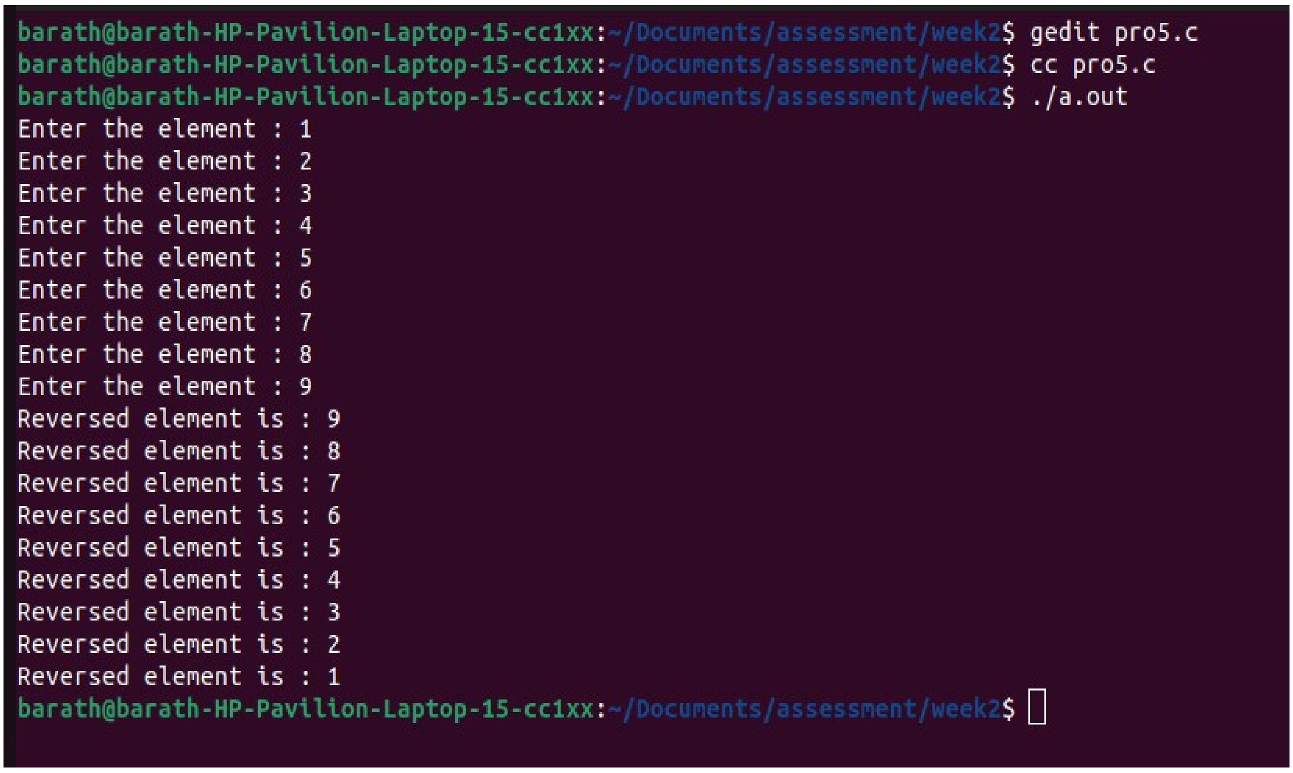
// Reverse the elements using stack

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	char data;
	struct node* next;
};
struct node *top=0, *newnode;
void push(){
	int a;
	printf("Enter the element : ");
	scanf("%d", &a);
	newnode=(struct node*)malloc(sizeof(struct node));
	newnode->data=a;
	newnode->next=top;
	top=newnode;
}
void pop(){
	struct node* temp;
	temp=top;
	if(top==NULL){
		printf("Stack empty");
	}
	else{
		printf("Reversed element is : %d \n", top->data);
		top=top->next;
		free(temp);
	}
}

int main()
{
	push();
	push();
	push();
	push();
	push();
	push();
	push();
	push();
	push();
	pop();
	pop();
	pop();
	pop();
	pop();
	pop();
	pop();
	pop();
	pop();
```

```
        return 0;
}
```

RESULT :



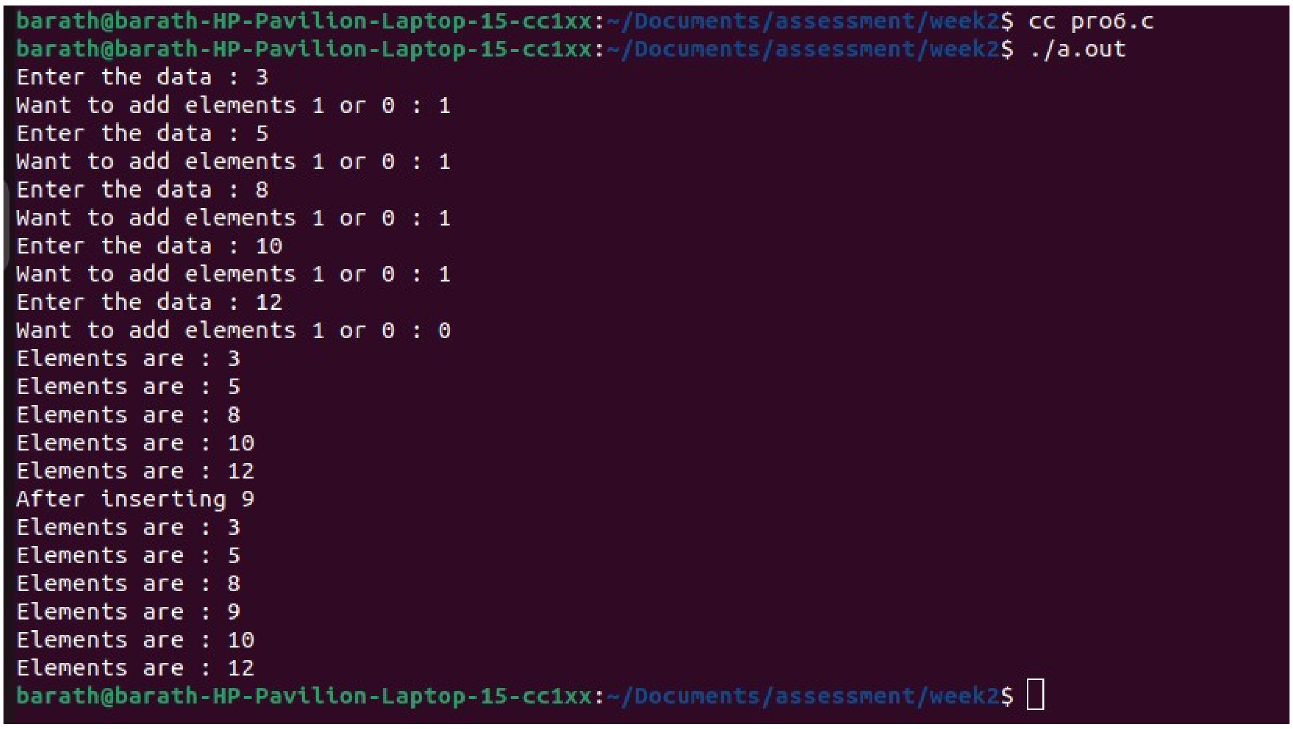
```
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ gedit pro5.c
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ cc pro5.c
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ ./a.out
Enter the element : 1
Enter the element : 2
Enter the element : 3
Enter the element : 4
Enter the element : 5
Enter the element : 6
Enter the element : 7
Enter the element : 8
Enter the element : 9
Reversed element is : 9
Reversed element is : 8
Reversed element is : 7
Reversed element is : 6
Reversed element is : 5
Reversed element is : 4
Reversed element is : 3
Reversed element is : 2
Reversed element is : 1
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ 
```

6.Insert element in sorted linked list.
// Insert element in sorted linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
	int data;
	struct node* next;
	struct node* prev;
};
struct node *newnode, *head=0;
void createDLL(){
	struct node *temp;
	int ch;
	while(ch){
	int a;
	printf("Enter the data : ");
	scanf("%d", &a);
	newnode=(struct node*)malloc(sizeof(struct node));
	newnode->data=a;
	newnode->prev=0;
	newnode->next=0;
	if(head == NULL){
		head=newnode;
		temp=newnode;
	}
	else{
		temp->next=newnode;
		newnode->prev=temp;
		temp=newnode;
	}
	printf("Want to add elements 1 or 0 : ");
	scanf("%d", &ch);
	}
}
void insert(int x){
	struct node *temp;
	temp=head;
	newnode=(struct node*)malloc(sizeof(struct node));
	newnode->data=x;
	newnode->prev=0;
	newnode->next=0;
	int i=1;
	if (head == NULL){
		printf("list empty");
	}
	else{
		do
		{
			if(temp->data < x){
				temp=temp->next;
			}
```

```
                        else
                        {
                                newnode->prev=temp->prev;
                                newnode->next=temp;
                                newnode->prev->next=newnode;
                                temp->prev=newnode;
                                i=0;
                        }
                }while(i);
        }
}
void display(){
        struct node *temp=head;
        if(head == NULL){
                printf(" List Empty ");
        }
        else{
                while(temp->next !=0){
                        printf("Elements are : %d\n", temp->data);
                        temp=temp->next;
                }
                printf("Elements are : %d\n", temp->data);
        }
}
int main(){
        createDLL();
        display();
        insert(9);
        printf("After inserting 9 \n");
        display();
}
```

RESULT :



```
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ cc pro6.c
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ ./a.out
Enter the data : 3
Want to add elements 1 or 0 : 1
Enter the data : 5
Want to add elements 1 or 0 : 1
Enter the data : 8
Want to add elements 1 or 0 : 1
Enter the data : 10
Want to add elements 1 or 0 : 1
Enter the data : 12
Want to add elements 1 or 0 : 0
Elements are : 3
Elements are : 5
Elements are : 8
Elements are : 10
Elements are : 12
After inserting 9
Elements are : 3
Elements are : 5
Elements are : 8
Elements are : 9
Elements are : 10
Elements are : 12
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$
```

7.Inseet/Delete and Count the elements in queue.

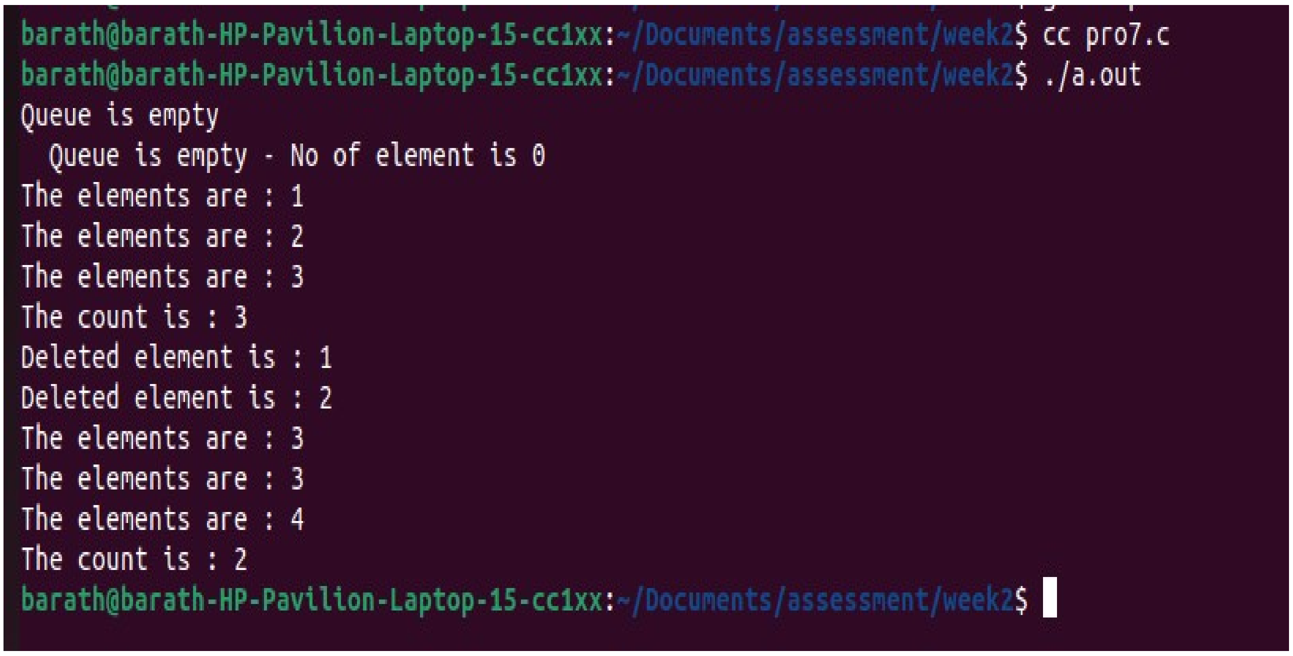
// insert/delete and count the elements in queue.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
        int data;
        struct node* next;
};
struct node *newnode, *front=0, *rear=0;
void isempty(){
        if(front == 0 || rear == 0){
                printf("Queue is empty \n ");
        }
        else{
                printf("Queue is not empty \n ");
        }
}
void enqueue(int x){
        newnode=(struct node*)malloc(sizeof(struct node));
        newnode->data=x;
        newnode->next=0;
        if(front ==0 || rear==0){
                front=newnode;
                rear=newnode;
        }
        else{
                rear->next=newnode;
                rear=newnode;
        }
}
void display(){
        struct node *temp;
        temp=front;
                while(temp != rear){
                        printf("The elements are : %d \n", temp->data);
                        temp=temp->next;
                }
                printf("The elements are : %d \n", temp->data);
}
void count(){
        struct node *temp;
        int count=0;
        temp=front;
        if(front == 0 || rear == 0){
                printf(" Queue is empty - No of element is 0 \n");
        }
        else{

                while(temp != rear){
                        count=count+1;
                        temp=temp->next;
                }
                count=count + 1;
                printf("The count is : %d\n", count);
        }
}
void dequeue(){
        struct node *temp;
        temp=front;
        if(front == rear)
        {
                printf("Queue is empty now ");

        }
        else{
                printf("Deleted element is : %d\n", front->data);
                front=front->next;
                free(temp);
        }
}
int main(){
        isempty();
        count();
        enqueue(1);
        enqueue(2);
        enqueue(3);
        display();
        count();
        dequeue();
        dequeue();
        display();
        enqueue(4);
        display();
        count();
}
```

RESULT:



```
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ cc pro7.c
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ ./a.out
Queue is empty
  Queue is empty - No of element is 0
The elements are : 1
The elements are : 2
The elements are : 3
The count is : 3
Deleted element is : 1
Deleted element is : 2
The elements are : 3
The elements are : 3
The elements are : 4
The count is : 2
barath@barath-HP-Pavilion-Laptop-15-cc1xx:~/Documents/assessment/week2$ 
```