**WEEK 3 DS LAB:**

**Queue Implementation:**

#include<stdio.h>

#include<stdlib.h>

#define size 3

int Q[size];

int rear=-1;

int front=-1;

int IsFull()

{

if(front==(rear+1)%size)

{

return 0;

}

else

{

return -1;

}

}

int IsEmpty()

{

if(front==-1 && rear==-1)

{

return 0;

}

else

{

return -1;

}

}

void Enqueue(int x)

{

int item;

if(IsFull()==0)

{

printf("Queue overflow \n");

return;

}

else

{

if(IsEmpty()==0)

{

front=0;

rear=0;

}

else

{

rear=(rear+1)%size;

}

Q[rear]=x;

}

}

int Dequeue()

{

int x;

if(IsEmpty()==0)

{

printf("Queue underflow \n");

}

else

{

if(front==rear)

{

x=Q[front];

front=-1;

rear=-1;

}

else

{

x=Q[front];

front=(front+1)%size;

}

return x;

}

}

void Display()

{

int i;

if(IsEmpty()==0)

{

printf("Queue is empty \n");

}

else

{

printf("Queue elements:\n");

for(i=front; i!=rear; i=(i+1)%size)

{

printf("%d \n",Q[i]);

}

printf("%d \n",Q[i]);

}

}

void main()

{

int choice,x,b;

while(1)

{

printf("1.Enqueue, 2.Dequeue, 3.Display, 4.exit \n");

printf("Enter your choice:");

scanf("%d", &choice);

switch(choice)

{

case 1:

printf("Enter the number to be inserted \n");

scanf("%d", &x);

Enqueue(x);

break;

case 2:

b=Dequeue();

printf("%d was removed from the queue \n",b);

break;

case 3:

Display();

break;

case 4:

exit(1);

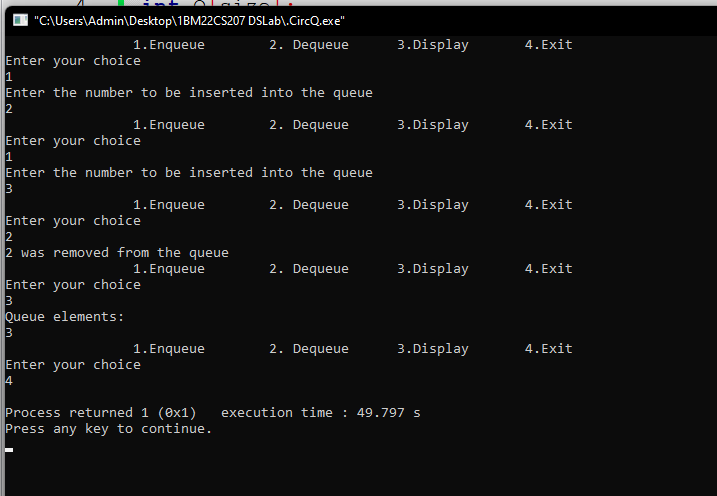
default:

printf("Invalid choice \n");

}

}

}

**OUTPUT:** 

**Circular Queue:**

#include <stdio.h>

#include <stdlib.h>

#define size 50

int Q[size];

int rear=-1;

int front=-1;

int IsFull()

{

if(front==(rear+1)%size)

{

return 0;

}

else

{

return -1;

}

}

int IsEmpty()

{

if(front==-1&&rear==-1)

{

return 0;

}

else

{

return -1;

}

}

void Enqueue(int x)

{

int item;

if(IsFull()==0)

{

printf("Queue Overflow\n");

}

else

{

if(IsEmpty()==0)

{

front=0;

rear=0;

}

else

{

rear=(rear+1)%size;

}

Q[rear]=x;

}

}

int Dequeue()

{

int x;

if(IsEmpty()==0)

{

printf("Queue underflow\n");

}

else

{

if(front==rear)

{

x=Q[front];

front=-1;

rear=-1;

}

else

{

x=Q[front];

front=(front+1)%size;

}

return x;

}

}

void Display()

{

int i;

if(IsEmpty()==0)

{

printf("Queue is empty\n");

}

else

{

printf("Queue elements:\n");

for(i=front;i!=rear;i=(i+1)%size)

{

printf("%d\n",Q[i]);

}

printf("%d \n",Q[i]);

}

}

void main()

{

int choice,x,b;

while(1)

{

printf("\t\t1.Enqueue\t 2. Dequeue\t 3.Display\t 4.Exit\n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter the number to be inserted into the queue\n");

scanf("%d",&x);

Enqueue(x);

break;

case 2:

b=Dequeue();

printf("%d was removed from the queue\n",b);

break;

case 3:

Display();

break;

case 4:

exit(1);

default:

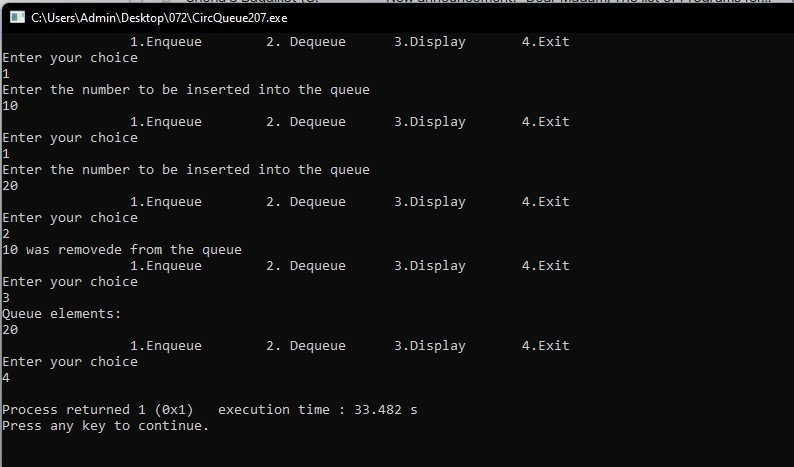
printf("Invalid input\n");

}

}

}

**OUTPUT:**



**Linked List:**

include <stdio.h>

#include <stdlib.h>

#include <conio.h>

typedef struct Node

{

int data;

struct Node\* next;

} Node;

Node\* head = NULL;

void push();

void append();

void insert();

void display();

void main() {

int choice;

while (1) {

printf("1. Insert at beginning\t 2. Insert at end\t 3. Insert at position\t 4. Display\t 5.Exit\t\n");

printf("Enter choice: \n ");

scanf("%d", &choice);

switch (choice)

{

case 1:

push();

break;

case 2:

append();

break;

case 3:

insert();

break;

case 4:

display();

break;

default:

printf("Exiting the program");

}

}

}

void push()

{

Node\* temp = (Node\*)malloc(sizeof(Node));

int new\_data;

printf("Enter data in the new node: ");

scanf("%d", &new\_data);

temp->data = new\_data;

temp->next = head;

head = temp;

}

void append()

{

Node\* temp = (Node\*)malloc(sizeof(Node));

int new\_data;

printf("Enter data in the new node: ");

scanf("%d", &new\_data);

temp->data = new\_data;

temp->next = NULL;

if (head == NULL) {

head = temp;

return;

}

Node\* temp1 = head;

while (temp1->next != NULL) {

temp1 = temp1->next;

}

temp1->next = temp;

}

void insert() {

Node\* temp = (Node\*)malloc(sizeof(Node));

int new\_data, pos;

printf("Enter data in the new node: ");

scanf("%d", &new\_data);

printf("Enter position of the new node: ");

scanf("%d", &pos);

temp->data = new\_data;

temp->next = NULL;

if (pos == 0) {

temp->next = head;

head = temp;

return;

}

Node\* temp1 = head;

while (pos--) {

temp1 = temp1->next;

}

Node\* temp2 = temp1->next;

temp->next = temp2;

temp1->next = temp;

}

void display()

{

Node\* temp1 = head;

while (temp1 != NULL) {

printf("%d -> ", temp1->data);

temp1 = temp1->next;

}

printf("NULL\n");

}

**OUTPUT:**

