DATA STRUCTURE PRACTICAL NO. :-08[A]

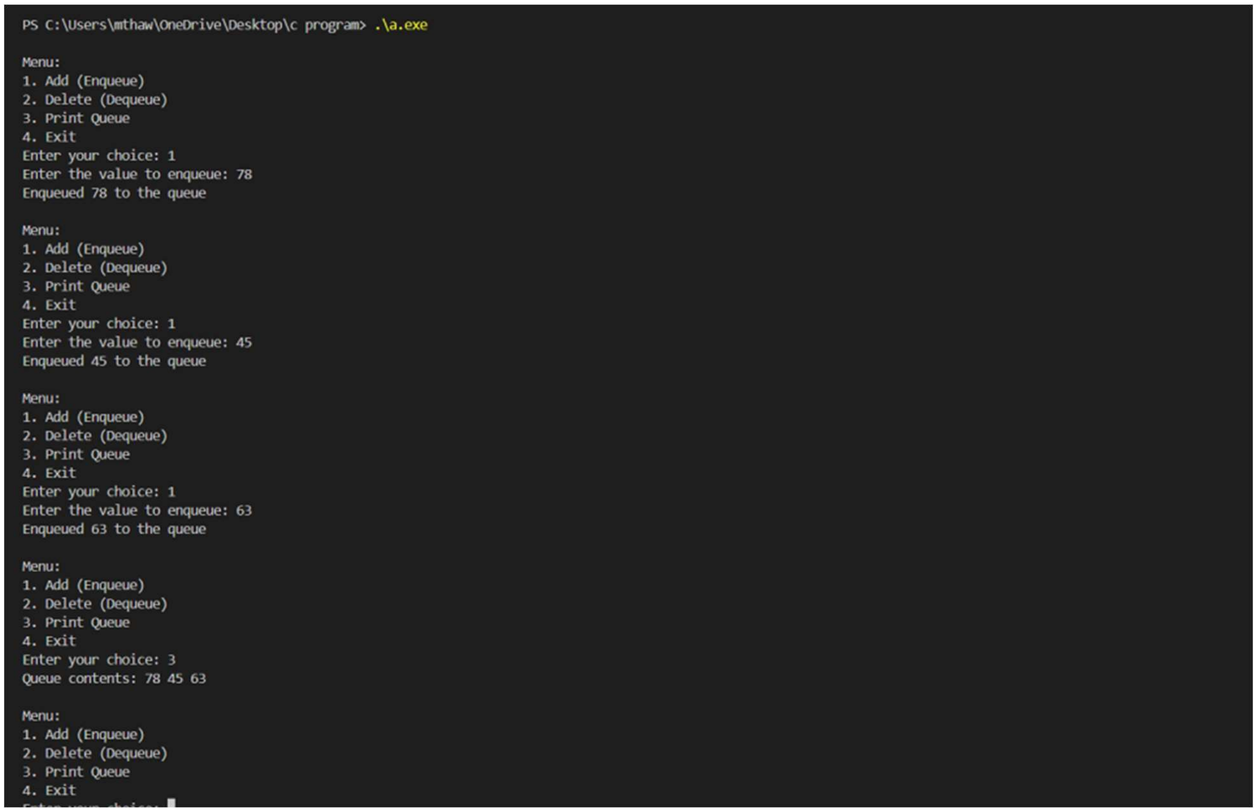
Aim : Implement a Queue and perform the Queue operations: Enqueue , Dequeue and Print using Menu Driver Program such as 1.Add, 2.Delete and 3. Print and 4. Exit.

PROGARM :-

```
#include<stdio.h>

int Queue[100];

int front = -1, rear = -1, data;

//FUNCTION FOR ENQUEUE

int enqueue(){

//Checking the queue is full or not

if(rear == 99){

printf("Sorry, The Queue is Overflow!\n");

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

{

printf("Enter the data:\t");

scanf("%d", &data);

//Checking the input element is first or not

front = 0;

rear = 0;

Queue[0] = data;

}else{

printf("Enter the data:\t");

scanf("%d", &data);

rear++;
```

```
Queue[rear] = data;
}
return 0;
}
//FUNCTION FOR DEQUEUE
int dequeue(){
//Checking the Queue is empty or not.
if(front == -1){
printf("The Queue is Empty to delete a element.\n");
}else if(front > rear){
//Checking all the element is deleted or not.
printf("The Queue is Empty to delete a element.\n");
front = -1;
rear = -1;
}else{
//Simply deleting the element from front.
printf("The deleting element is %d\n", Queue[front]);
front++;
}
return 0;
}
void display(){
if(front == -1 || front > rear){
//Checking the queue is empty or not.
```

```
printf("The Queue is empty so, can not print the element.\n");
}else{
//printing the elements in the Queue
printf("The element in the Queue are:\t");
for(int i = front; i <= rear; i++){
printf("%d\t", Queue[i]);
}
printf("\n");


}
}
//MAIN FUNCTION
int main(){
int choice;
printf("Queue Implementation\n");
printf("Choices\n1.Enqueue\t2.Dequeue\t3.Print\t4.Exit\n");
do
{
printf("Enter a valid choice\n");
scanf("%d", &choice);
switch (choice)
{
case 1:
enqueue();
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 4:
printf("You exited the Program successfully.");
break;
default:
printf("Please enter a valid choice as mention!\n");
break;
}
} while (choice != 4);
return 0;
}
```



```
PS C:\Users\mthaw\OneDrive\Desktop\c program> .\a.exe

Menu:
1. Add (Enqueue)
2. Delete (Dequeue)
3. Print Queue
4. Exit
Enter your choice: 1
Enter the value to enqueue: 78
Enqueued 78 to the queue

Menu:
1. Add (Enqueue)
2. Delete (Dequeue)
3. Print Queue
4. Exit
Enter your choice: 1
Enter the value to enqueue: 45
Enqueued 45 to the queue

Menu:
1. Add (Enqueue)
2. Delete (Dequeue)
3. Print Queue
4. Exit
Enter your choice: 1
Enter the value to enqueue: 63
Enqueued 63 to the queue

Menu:
1. Add (Enqueue)
2. Delete (Dequeue)
3. Print Queue
4. Exit
Enter your choice: 3
Queue contents: 78 45 63

Menu:
1. Add (Enqueue)
2. Delete (Dequeue)
3. Print Queue
4. Exit
```

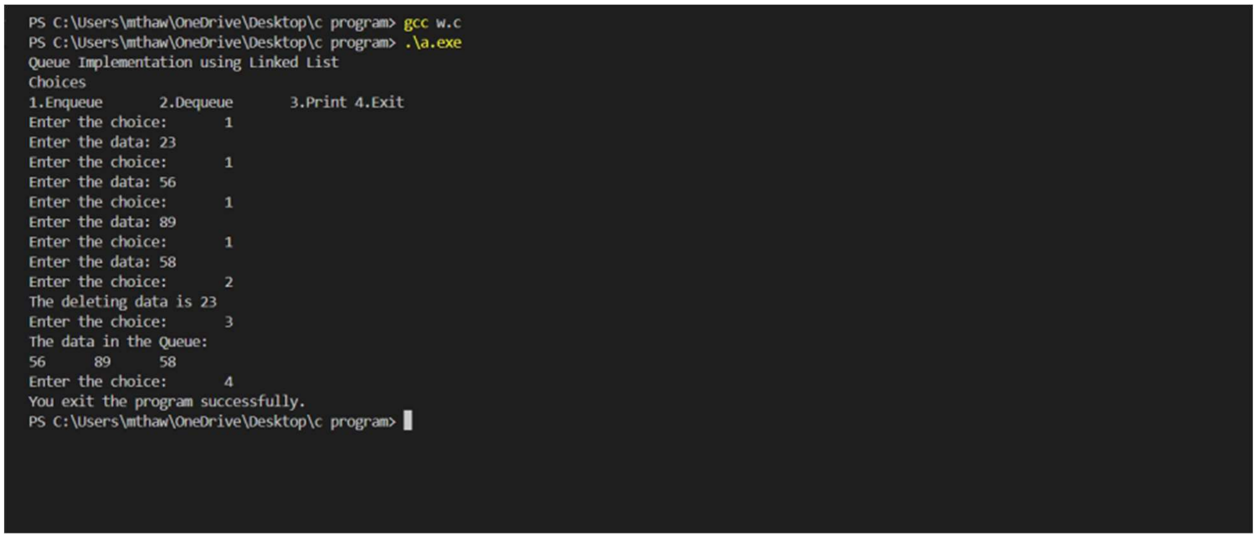
DATA STRUCTURE PRACTICAL NO. :-08[B]

Aim : : Implement a Queue using Linked List and perform the Queue operations: Enqueue, Dequeue and Print using Menu Driver Program such as 1.Add, 2.Delete and 3.Print and 4. Exit.

PROGRAM :-

```
//Queue Implementation using linked list

#include<stdio.h>

#include<stdlib.h>


//Structure of the node

struct node{

    int data;
    struct node* next;
};
int data;
struct node* front = NULL;
struct node* rear = NULL;

//Inserting data in queue.(Enqueue function):
int enqueue(){
    //Creating the node first
    struct node* p;
    p = (struct node*)malloc(sizeof(struct node));
    if(p == NULL){
        //Checking the queue is overflow or not
        printf("The Queue is overflow\n");
    }
    printf("Enter the data:\t");
    scanf("%d", &p->data);
    p->next = NULL; // Initialize new node's next to NULL

    if (front == NULL && rear == NULL)
    {

        // First element in queue
        front = rear = p;
    }
    else
    {
        // Add to the end of the queue
        rear->next = p;
        rear = p;
    }

    return 0;
}

// Deleting data in queue.(Dequeue function):
int dequeue(){
    struct node* p;
    if(front == NULL && rear == NULL){
        printf("The Queue is underflow\n");
    }
    else
    {
        struct node *p = front;
        printf("The deleting data is %d\n", front->data);
        front = front->next;

        if (front == NULL)
        {
            // If queue becomes empty, update rear to NULL
            rear = NULL;
        }
        free(p);

    }

    return 0;
}

void display(){
    struct node* display;
    display = front;
    if(front == NULL){
        printf("The Queue is empty can not print the element.\n\n");
    }else{
    printf("The data in the Queue:\t\n");
    while(display != NULL){
        printf("%d\t", display -> data);
        display = display -> next;
    }
    printf("\n" );
    }
```

```
}

int main(){
    int choice;
    printf("Queue Implementation using Linked List\n");
    printf("Choices\n1.Enqueue\t2.Dequeue\t3.Print\t4.Exit\n");
    do
    {   printf("Enter the choice:\t");
        scanf("%d",&choice);

        switch (choice)
        {
        case 1:

            enqueue();
            break;
        case 2:
            dequeue();
            break;
        case 3:
            display();
            break;
        case 4:
            printf("You exit the program successfully.\n");
            break;
        default:
        printf("Please enter valid choice as mention\n");
            break;
        }

    } while (choice != 4);

    return 0;
}
```



```
PS C:\Users\mthaw\OneDrive\Desktop\c program> gcc w.c
PS C:\Users\mthaw\OneDrive\Desktop\c program> .\a.exe
Queue Implementation using Linked List
Choices
1.Enqueue       2.Dequeue       3.Print 4.Exit
Enter the choice:       1
Enter the data: 23
Enter the choice:       1
Enter the data: 56
Enter the choice:       1
Enter the data: 89
Enter the choice:       1
Enter the data: 58
Enter the choice:       2
The deleting data is 23
Enter the choice:       3
The data in the Queue:
56      89      58
Enter the choice:       4
You exit the program successfully.
PS C:\Users\mthaw\OneDrive\Desktop\c program>
```

DATA STRUCTURE PRACTICAL NO. :-08[C]

Aim :- Implement a Circular Queue and perform the Queue operations: Enqueue, Dequeue and Print using Menu Driver Program such as 1.Add, 2.Delete and 3.Print and 4.Exit.

PROGRAM:-

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

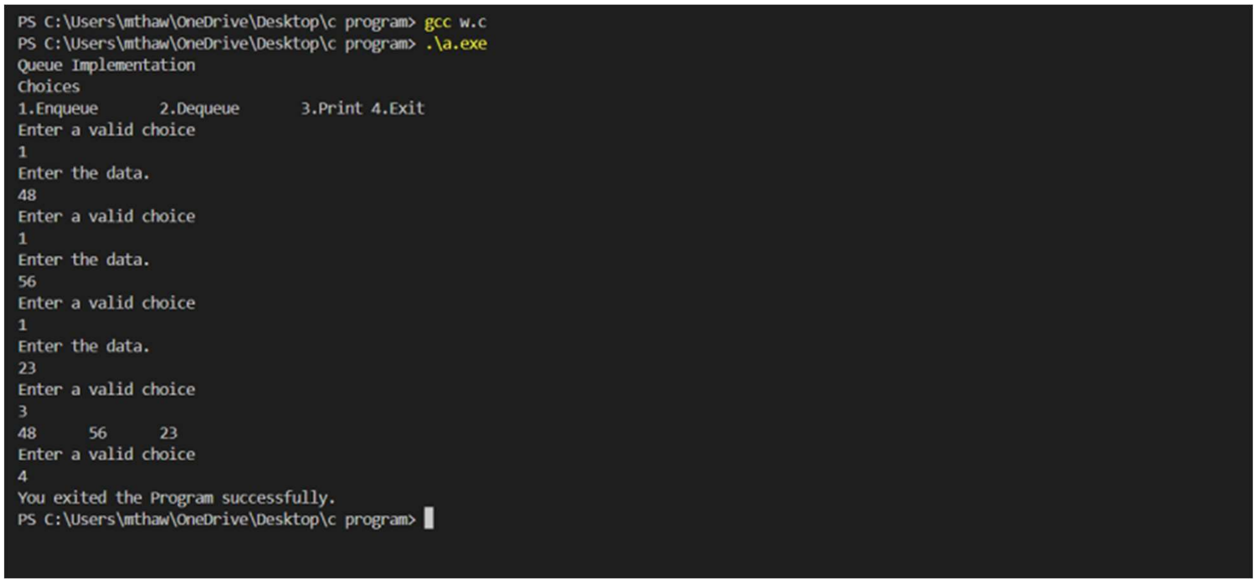
```
// Creating array Globaly
int Queue[5];
int front = -1, rear = -1, data;

// FUNCTION FOR ENQUEUE
int enqueue()
{
   if((rear + 1) % 5 == front){
      printf("The Queue is Overflow.\n");
   }else if(front == -1 && rear == -1){
      front = 0;
      rear = 0;
      printf("Enter the data.\n");
      scanf("%d", &data);
      Queue[rear] = data;
   }else{
      printf("Enter the data.\n");
      scanf("%d", &data);
      rear = (rear + 1) % 5;
      Queue[rear] = data;
   }
   return 0;
}
```

```
// FUNCTION FOR DEQUEUE
int dequeue()

{

   if(front == -1 && rear == -1 ){
      printf("The Queue is Underflow.\n");
   }else if(front == rear){
      printf("The Queue is Underflow.\n");
      front = rear = -1;
   }else{
      printf("The deleting element is %d.\n", Queue[front]);
      front = (front + 1) % 5;
   }
   return 0;
}

void display()
{
   if (front == -1)
   {
      // Checking the queue is empty or not.
      printf("The Queue is empty so, can not print the element.\n");
```

```
    }
    else
    {
        // printing the elements in the Queue
        int i = front;
        while (1)
        {
            printf("%d\t", Queue[i]);
            if (i == rear)
                break;       // Stop when we reach the rear
            i = (i + 1) % 5; // Move to the next index in circular manner

        }
        printf("\n");
    }
}

// MAIN FUNCTION
int main()
{
    int choice;
    printf("Queue Implementation\n");
    printf("Choices\n1.Enqueue\t2.Dequeue\t3.Print\t4.Exit\n");
    do
```

```
{
   printf("Enter a valid choice\n");
   scanf("%d", &choice);

   switch (choice)
   {
   case 1:
      enqueue();
      break;
   case 2:
      dequeue();
      break;

   case 3:
      display();
      break;

   case 4:
      printf("You exited the Program successfully.");

      break;

   default:
      printf("Please enter a valid choice as mention!\n");
```

```
            break;
        }
    } while (choice != 4);

    return 0;
}
```



```
PS C:\Users\mthaw\OneDrive\Desktop\c program> gcc w.c
PS C:\Users\mthaw\OneDrive\Desktop\c program> .\a.exe
Queue Implementation
Choices
1.Enqueue       2.Dequeue       3.Print 4.Exit
Enter a valid choice
1
Enter the data.
48
Enter a valid choice
1
Enter the data.
56
Enter a valid choice
1
Enter the data.
23
Enter a valid choice
3
48      56      23
Enter a valid choice
4
You exited the Program successfully.
PS C:\Users\mthaw\OneDrive\Desktop\c program>
```

GITHUB LINK:-

https://github.com/sidheshwar2005/Data_strucutre_practical.git