1. **Write a menu driven program in C to perform Linear Queue operations (Enqueue,Dequeue).**

**Program: prg3a.c**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 3

void Enqueue();

void Dequeue();

void display();

int arr[MAX\_SIZE], FRONT = -1, REAR = -1;

int main()

{

int choice;

while (1)

{

printf("\n\n\t\t--------- Linear Queue Operation ----------\n");

printf(" 1. Enqueue\n 2. Dequeue\n 0. Exit\n");

printf("\nEnter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 0:

printf("\n\tTHANK YOU FOR USING THE PROGRAM\n");

exit(0);

case 1:

Enqueue();

display();

break;

case 2:

Dequeue();

display();

break;

default:

printf("\n\tERROR.. Wrong Choice !!!\t");

}

printf("\n\nPress Enter to continue.... ");

getchar();

}

return 0;

}

void Enqueue()

{

int data;

if (REAR == MAX\_SIZE - 1)

{

printf("\n\tQueue is full..! Can't Insert new element\n\n");

return;

}

else

{

if (FRONT == -1)

{

FRONT = 0;

}

printf("\nEnter the data : ");

scanf("%d", &data);

REAR = REAR + 1;

arr[REAR] = data;

}

}

void Dequeue()

{

if (FRONT == -1 || FRONT > REAR)

{

printf("\n\tQueue is empty..! Can't delete an element\n\n");

return;

}

else

{

printf("\n\tDeleted : %d\n", arr[FRONT]);

FRONT = FRONT + 1;

}

}

void display()

{

int i;

if (FRONT == -1)

{

printf("\n\tQueue is empty..!\n\n");

return;

}

else

{

printf("\nThe Queue is : \n");

for (i = FRONT; i <= REAR; i++)

{

if (i == FRONT)

{

printf(" FRONT (%d) --> |", FRONT);

}

printf(" %d |", arr[i]);

if (i == REAR)

{

printf(" <-- REAR (%d)", REAR);

}

}

}

}

**OUTPUT:**

Enter your choice : 2

Deleted : 30

The Queue is :

FRONT (3) --> <-- REAR (2)

Press Enter to continue....

Enter your choice : 2

Queue is empty..! Can't delete an element

The Queue is :

FRONT (3) --> <-- REAR (2)

Press Enter to continue....

Enter your choice : 0

THANK YOU FOR USING THE PROGRAM

--------- Linear Queue Operation ----------

1. Enqueue

2. Dequeue

0. Exit

Enter your choice : 1

Enter the data : 10

The Queue is :

FRONT (0) --> | 10 | <-- REAR (0)

Press Enter to continue....

Enter your choice : 1

Enter the data : 20

The Queue is :

FRONT (0) --> | 10 | 20 | <-- REAR (1)

Press Enter to continue.…

Enter your choice : 1

Enter the data : 30

The Queue is :

FRONT (0) --> | 10 | 20 | 30 | <-- REAR (2)

Press Enter to continue....

Enter your choice : 1

Queue is full..! Can't Insert new element

the Queue is :

FRONT (0) --> | 10 | 20 | 30 | <-- REAR (2)

Press Enter to continue.…

Enter your choice : 2

Deleted : 10

The Queue is :

FRONT (1) --> | 20 | 30 | <-- REAR (2)

Press Enter to continue....

Enter your choice : 2

Deleted : 20

The Queue is :

FRONT (2) --> | 30 | <-- REAR (2)

Press Enter to continue....

1. **Write a menu driven program in C to perform Circular Queue operations (Enqueue, Dequeue).**

**Program: prg3b.c**

#include <stdio.h>

#include <stdlib.h>

#define MAX 3

int queue[MAX];

int FRONT = -1, REAR = -1;

void Enqueue();

void Dequeue();

void display();

int main()

{

int choice;

while (1)

{

printf("\n\n\t\t--------- Circular Queue Operation ----------\n");

printf(" 1. Enqueue\n 2. Dequeue\n 0. Exit\n");

printf("\nEnter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 0:

printf("\n\tTHANK YOU FOR USING THE PROGRAM\n");

exit(0);

case 1:

Enqueue();

display();

break;

case 2:

Dequeue();

display();

break;

default:

printf("\n\tERROR.. Wrong Choice !!!\t");

}

printf("\n\nPress Enter to continue.... ");

}

return 0;

}

void Enqueue()

{

int data;

if ((FRONT == 0 && REAR == MAX - 1) || (FRONT == REAR + 1))

{

printf("\n\tQueue is full..! Can't Insert new element\n\n");

return;

}

else

{

if (FRONT == -1)

{

FRONT = 0;

REAR = 0;

}

else

{

if (REAR == MAX - 1)

REAR = 0;

else

REAR = REAR + 1;

}

printf("\nEnter the element : ");

scanf("%d", &data);

queue[REAR] = data;

}

}

void Dequeue()

{

if (FRONT == -1)

{

printf("\n\tQueue is empty..! Can't delete element\n\n");

return;

}

else

{

printf("\n\tDeleted element is : %d\n", queue[FRONT]);

if (FRONT == REAR)

{

FRONT = -1;

REAR = -1;

}

else

{

if (FRONT == MAX - 1)

FRONT = 0;

else

FRONT = FRONT + 1;

}

}

}

void display()

{

int i;

if (FRONT == -1)

{

printf("\n\tCircular Queue is empty\n\n");

return;

}

printf("\nFRONT (%d) -> |", FRONT);

if (FRONT <= REAR)

{

for (i = FRONT; i <= REAR; i++)

{

printf(" %d |", queue[i]);

}

}

else

{

for (i = FRONT; i < MAX; i++)

{

printf(" %d |", queue[i]);

}

for (i = 0; i <= REAR; i++)

{

printf(" %d |", queue[i]);

}

}

printf(" <- REAR (%d)\n", REAR);

}

**OUTPUT:**

Deleted element is : 20

FRONT (2) -> | 30 | <- REAR (2)

Press Enter to continue....

Enter your choice : 2

Deleted element is : 30

Circular Queue is empty

Press Enter to continue....

Enter your choice : 0

THANK YOU FOR USING THE PROGRAM

--------- Circular Queue Operation ----------

1. Enqueue

2. Dequeue

0. Exit

Enter your choice : 1

Enter the element : 10

FRONT (0) -> | 10 | <- REAR (0)

Press Enter to continue....

Enter your choice : 1

Enter the element : 20

FRONT (0) -> | 10 | 20 | <- REAR (1)

Press Enter to continue....

Enter your choice : 1

Enter the element : 30

FRONT (0) -> | 10 | 20 | 30 | <- REAR (2)

Press Enter to continue....

Enter your choice : 1

Queue is full..! Can't Insert new element

FRONT (0) -> | 10 | 20 | 30 | <- REAR (2)

Press Enter to continue....

Enter your choice : 2

Deleted element is : 10

FRONT (1) -> | 20 | 30 | <- REAR (2)

Press Enter to continue....

Enter your choice : 2