**PROGRAM:**

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

typedef struct Node

{

int data;

struct Node\* next;

} Node;

Node\* createNode(int new\_data)

{

Node\* new\_node = (Node\*)malloc(sizeof(Node));new\_node->data = new\_data;

new\_node->next = NULL;

return new\_node;

}

typedef struct Queue

{

Node \*front, \*rear;

} Queue;

Queue\* createQueue()

{

Queue\* q = (Queue\*)malloc(sizeof(Queue));

q->front = q->rear = NULL;

return q;

}

int isEmpty(Queue\* q)

{

return q->front == NULL && q->rear == NULL;

}

void enqueue(Queue\* q, int new\_data)

{

Node\* new\_node = createNode(new\_data);

if (q->rear == NULL)

{

q->front = q->rear = new\_node;

return;

}

q->rear->next = new\_node;

q->rear = new\_node;

}

void dequeue(Queue\* q)

{

if (isEmpty(q))

{

printf("Queue Underflow\n");

return;

}

Node\* temp = q->front;

q->front = q->front->next;

if (q->front == NULL) q->rear = NULL;

free(temp);

}

int getFront(Queue\* q)

{

if (isEmpty(q))

{

printf("Queue is empty\n");

return INT\_MIN;

}

return q->front->data;

}

int getRear(Queue\* q)

{

if (isEmpty(q))

{

printf("Queue is empty\n");

return INT\_MIN;

}

return q->rear->data;

}

int main()

{

Queue\* q = createQueue();

int choice, value;

while (1)

{

printf("\nQueue Operations:\n");

printf("1. Enqueue\t 2. Dequeue\t 3. Get Front\n4. Get Rear\t

5. Exit\n Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("Enter value to enqueue: ");

scanf("%d", &value);enqueue(q, value);

printf("Enqueued %d\n", value);

break;

case 2:

dequeue(q);

printf("Dequeued from queue\n");

break;

case 3:

value = getFront(q);

if (value != INT\_MIN)

{

printf("Front element: %d\n", value);

}

break;

case 4:

value = getRear(q);

if (value != INT\_MIN)

{

printf("Rear element: %d\n", value);

}

break;

case 5:

printf("Program executed successfully. \n");

exit(0);

default:

printf("Invalid choice. Try again.\n");

}

}

return 0;

}

**OUTPUT:**

PS D:\C Data Structure\10. Queue Implementation> gcc Queue.c

PS D:\C Data Structure\10. Queue Implementation> ./a

Queue Operations:

1. Enqueue 2. Dequeue 3. Get Front

4. Get Rear 5. Exit

Enter your choice: 1

Enter value to enqueue: 13

Enqueued 13

Queue Operations:

1. Enqueue 2. Dequeue 3. Get Front

4. Get Rear 5. Exit

Enter your choice: 1

Enter value to enqueue: 21

Enqueued 21

Queue Operations:

1. Enqueue 2. Dequeue 3. Get Front

4. Get Rear 5. Exit

Enter your choice: 4

Rear element: 21

Queue Operations:

1. Enqueue 2. Dequeue 3. Get Front

4. Get Rear 5. Exit

Enter your choice: 3

Front element: 13

Queue Operations:

1. Enqueue 2. Dequeue 3. Get Front

4. Get Rear 5. Exit

Enter your choice: 2

Dequeued from queue

Queue Operations:

1. Enqueue 2. Dequeue 3. Get Front

4. Get Rear 5. Exit

Enter your choice: 4

Rear element: 21

Queue Operations:

1. Enqueue 2. Dequeue 3. Get Front

4. Get Rear 5. Exit

Enter your choice: 5

Program executed successfully.

PS D:\C Data Structure\10. Queue Implementation>