

# LINKED LIST INSERTION OPERATIONS

## PROGRAM:

```
Start here X Double ended queue.c X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX 5
5
6 int dequeue[MAX];
7 int front = -1, rear = -1;
8
9 // Insert at the front end
10 void insertFront(int value) {
11     if ((front == 0 && rear == MAX - 1) || (front == rear + 1)) {
12         printf("Dequeue is full! Cannot insert.");
13         return;
14     }
15     if (front == -1) { // First element insertion
16         front = rear = 0;
17     } else if (front == 0) {
18         front = MAX - 1;
19     } else {
20         front -= 1;
21     }
22     dequeue[front] = value;
23     printf("%d inserted at front.", value);
24 }
25
26 // Insert at the rear end
27 void insertRear(int value) {
28     if ((front == 0 && rear == MAX - 1) || (front == rear + 1)) {
29         printf("Dequeue is full! Cannot insert.");
30         return;
31     }
32     if (front == -1) { // First element insertion
33         front = rear = 0;
34     } else if (rear == MAX - 1) {
35         rear = 0;
36     } else {
```

Start here X Double ended queue.c X

```
36     } else {
37         rear += 1;
38     }
39     dequeue[rear] = value;
40     printf("%d inserted at rear.", value);
41 }
42
43 // Delete from the front end
44 void deleteFront() {
45     if (front == -1) {
46         printf("Dequeue is empty! Cannot delete.");
47         return;
48     }
49     printf("%d deleted from front.", dequeue[front]);
50     if (front == rear) { // Only one element
51         front = rear = -1;
52     } else if (front == MAX - 1) {
53         front = 0;
54     } else {
55         front++;
56     }
57 }
58
59 // Delete from the rear end
60 void deleteRear() {
61     if (front == -1) {
62         printf("Dequeue is empty! Cannot delete.");
63         return;
64     }
65     printf("%d deleted from rear.", dequeue[rear]);
66     if (front == rear) { // Only one element
67         front = rear = -1;
68     } else if (rear == 0) {
69         rear = MAX - 1;
70     } else {
71         rear--;
72     }
73 }
```

```
71             rear--;
72         }
73     }
74
75     // Display the dequeue
76     void display() {
77         if (front == -1) {
78             printf("Dequeue is empty!");
79             return;
80         }
81         printf("Dequeue elements are: ");
82         int i = front;
83         while (1) {
84             printf("%d ", dequeue[i]);
85             if (i == rear)
86                 break;
87             i = (i + 1) % MAX;
88         }
89         printf(" ");
90     }
91
92     int main() {
93         int choice, value;
94
95         while (1) {
96             printf("---- DOUBLE ENDED QUEUE (DEQUE) ----");
97             printf("1. Insert at front");
98             printf("2. Insert at rear");
99             printf("3. Delete front");
100            printf("4. Delete rear");
101            printf("5. Display queue");
102            printf("6. Exit");
103            printf("Enter your choice: ");
104            scanf("%d", &choice);
105
106            switch (choice) {
```

```
100     printf("4. Delete rear");
101     printf("5. Display queue");
102     printf("6. Exit");
103     printf("Enter your choice: ");
104     scanf("%d", &choice);
105
106     switch (choice) {
107         case 1:
108             printf("Enter value to insert at front: ");
109             scanf("%d", &value);
110             insertFront(value);
111             break;
112         case 2:
113             printf("Enter value to insert at rear: ");
114             scanf("%d", &value);
115             insertRear(value);
116             break;
117         case 3:
118             deleteFront();
119             break;
120         case 4:
121             deleteRear();
122             break;
123         case 5:
124             display();
125             break;
126         case 6:
127             exit(0);
128         default:
129             printf("Invalid choice! Try again.");
130     }
131 }
132
133
134 }
```

```
--> C:\Users\rupti\OneDrive\Desktop> x + v
--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 1
Enter data to insert: 10
Node inserted at beginning.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 2
e: 2
Enter data to insert: 20
Node inserted at end.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 2
Enter data to insert: 30
Node inserted at end.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 3
Enter data to insert: 25
Enter position: 3
Node inserted at position 3.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 4
Linked List elements: 10 -> 20 -> 25 -> 30 -> NULL--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 5
Enter data to insert: 5
Node inserted at beginning.--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 4
e: 4
Linked List elements: 5 -> 10 -> 20 -> 25 -> 30 -> NULL--- Singly Linked List Operations ---1. Insert at Beginning2. Insert at End3. Insert at Position4. Display5. ExitEnter your choice: 5
splay5. ExitEnter your choice: 5
Exiting...
Process returned 0 (0x0)  execution time : 99.988 s
Press any key to continue.
```