```
Q1. Write a menu driven program to implement queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty using linked list
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *ptr;
} * front, *rear, *temp, *front1;
int Peek();
void enq(int data);
void deq();
void empty();
void display();
void create();
void main()
{
   int no, ch, e;
   create();
   while (1)
   {
      printf("\n 1 - Enque");
      printf("\n 2 - Deque");
      printf("\n 3 - Peek");
      printf("\n 4 - Empty");
      printf("\n 5 - Display");
      printf("\n 6 - Exit");
      printf("\n Enter choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         printf("Enter data : ");
         scanf("%d", &no);
         enq(no);
         break;
      case 2:
         deq();
         break;
      case 3:
         e = Peek();
         if (e != 0)
            printf("Front element : %d", e);
         else
```

```
            printf("\n No front element in Queue as queue is empty");
         break;
      case 4:
         empty();
         break;
      case 5:
         display();
         break;
      case 6:
         exit(0);
      default:
         printf("Wrong choice, Please enter correct choice  ");
         break;
      }
   }
}
void create()
{
   front = rear = NULL;
}
void enq(int data)
{
   if (rear == NULL)
   {
      rear = (struct node *)malloc(1 * sizeof(struct node));
      rear->ptr = NULL;
      rear->info = data;
      front = rear;
   }
   else
   {
      temp = (struct node *)malloc(1 * sizeof(struct node));
      rear->ptr = temp;
      temp->info = data;
      temp->ptr = NULL;
      rear = temp;
   }
}
void display()
{
   front1 = front;

   if ((front1 == NULL) && (rear == NULL))
   {
      printf("Queue is empty");
```

```
        return;
    }
    while (front1 != rear)
    {
        printf("%d ", front1->info);
        front1 = front1->ptr;
    }
    if (front1 == rear)
        printf("%d", front1->info);
}
void deq()
{
    front1 = front;

    if (front1 == NULL)
    {
        printf("\n Error: Trying to display elements from empty queue");
        return;
    }
    else if (front1->ptr != NULL)
    {
        front1 = front1->ptr;
        printf("\n Dequed value : %d", front->info);
        free(front);
        front = front1;
    }
    else
    {
        printf("\n Dequed value : %d", front->info);
        free(front);
        front = NULL;
        rear = NULL;
    }
}
int Peek()
{
    if ((front != NULL) && (rear != NULL))
        return (front->info);
    else
        return 0;
}
void empty()
{
    if ((front == NULL) && (rear == NULL))
        printf("\n Queue empty");
```

```
  else
     printf("Queue not empty");
}
```

OUTPUT

```
  1 - Enque
  2 - Deque
  3 - Peek
  4 - Empty
  5 - Display
  6 - Exit
  Enter choice : 1
 Enter data : 1

  Enter choice : 1
 Enter data : 2

  Enter choice : 1
 Enter data : 3

  Enter choice : 1
 Enter data : 4

  Enter choice : 1
 Enter data : 5

  Enter choice : 5
 1 2 3 4 5
  Enter choice : 2

  Dequed value : 1
  Enter choice : 5
 2 3 4 5
  Enter choice : 3
 Front element : 2
  Enter choice : 6
```

Q2. Write a menu driven program to implement circular queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty, IsFull using static array.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

int cqueue_arr[MAX];
int front = -1;
int rear = -1;
void display();
void insert(int item);
int del();
int peek();
int isEmpty();
int isFull();
int main()
{
   int choice, item;
   while (1)
   {
      printf("\n 1 - Enque");
```

```
        printf("\n 2 - Deque");
        printf("\n 3 - Peek");
        printf("\n 4 - Display");
        printf("\n 5 - Exit");
        printf("\nEnter your choice : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("\nInput the element for insertion : ");
            scanf("%d", &item);
            insert(item);
            break;
        case 2:
            printf("\nElement deleted is : %d\n", del());
            break;
        case 3:
            printf("\nElement at the front is  : %d\n", peek());
            break;
        case 4:
            display();
            break;
        case 5:
            exit(1);
        default:
            printf("\nWrong choice\n");
        }
    }
    return 0;
}
void insert(int item)
{
    if (isFull())
    {
        printf("\nQueue Overflow\n");
        return;
    }
    if (front == -1)
        front = 0;

    if (rear == MAX - 1)
        rear = 0;
    else
        rear = rear + 1;
```

```
    cqueue_arr[rear] = item;
}
int del()
{
    int item;
    if (isEmpty())
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    item = cqueue_arr[front];
    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
    else if (front == MAX - 1)
        front = 0;
    else
        front = front + 1;
    return item;
}
int isEmpty()
{
    if (front == -1)
        return 1;
    else
        return 0;
}
int isFull()
{
    if ((front == 0 && rear == MAX - 1) || (front == rear + 1))
        return 1;
    else
        return 0;
}
int peek()
{
    if (isEmpty())
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    return cqueue_arr[front];
}
```

```c
void display()
{
  int i;
  if (isEmpty())
  {
    printf("\nQueue is empty\n");
    return;
  }
  printf("\nQueue elements :\n");
  i = front;
  if (front <= rear)
  {
    while (i <= rear)
      printf("%d ", cqueue_arr[i++]);
  }
  else
  {
    while (i <= MAX - 1)
      printf("%d ", cqueue_arr[i++]);
    i = 0;
    while (i <= rear)
      printf("%d ", cqueue_arr[i++]);
  }
  printf("\n");
}
```

OUTPUT

```
1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Input the element for insertion : 1

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Input the element for insertion : 2
```

```
 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Input the element for insertion : 3

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Input the element for insertion : 4

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 4

Queue elements :
1 2 3 4

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 2

Element deleted is : 1

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 3

Element at the front is  : 2
```

```
 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 5
```

Q3. Write a menu driven program to implement Deques (both Input-restricted and Output-restricted) operations such as Enqueue,Dequeue, Peek, Display of elements, IsEmpty, IsFull using static array.

```
#include <stdio.h>
#define MAX 5
int deque_arr[MAX];
int left = -1;
int right = -1;
void Enqueue_right()
{
   int added_item;
   if ((left == 0 && right == MAX - 1) || (left == right + 1))
   {
      printf("Queue Overflow\n");
      return;
   }
   if (left == -1)
   {
      left = 0;
      right = 0;
   }
   else if (right == MAX - 1)
      right = 0;
   else
      right = right + 1;
   printf("Input the element for adding in queue : ");
   scanf("%d", &added_item);
   deque_arr[right] = added_item;
}
void Enqueue_left()
{
   int added_item;
   if ((left == 0 && right == MAX - 1) || (left == right + 1))
   {
      printf("Queue Overflow \n");
      return;
   }
   if (left == -1)
```

```
    {
        left = 0;
        right = 0;
    }
    else if (left == 0)
        left = MAX - 1;
    else
        left = left - 1;
    printf("Input the element for adding in queue : ");
    scanf("%d", &added_item);
    deque_arr[left] = added_item;
}
void delete_left()
{
    if (left == -1)
    {
        printf("Queue Underflow\n");
        return;
    }
    printf("Element deleted from queue is : %d\n", deque_arr[left]);
    if (left == right)
    {
        left = -1;
        right = -1;
    }
    else if (left == MAX - 1)
        left = 0;
    else
        left = left + 1;
}
void delete_right()
{
    if (left == -1)
    {
        printf("Queue Underflow\n");
        return;
    }
    printf("Element deleted from queue is : %d\n", deque_arr[right]);
    if (left == right)
    {
        left = -1;
        right = -1;
    }
    else if (right == 0)
        right = MAX - 1;
```

```
    else
        right = right - 1;
}
void display_queue()
{
    int front_pos = left, rear_pos = right;
    if (left == -1)
    {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements :\n");
    if (front_pos <= rear_pos)
    {
        while (front_pos <= rear_pos)
        {
            printf("%d ", deque_arr[front_pos]);
            front_pos++;
        }
    }
    else
    {
        while (front_pos <= MAX - 1)
        {
            printf("%d ", deque_arr[front_pos]);
            front_pos++;
        }
        front_pos = 0;
        while (front_pos <= rear_pos)
        {
            printf("%d ", deque_arr[front_pos]);
            front_pos++;
        }
    }
    printf("\n");
}
void input_que()
{
    int choice;
    do
    {
        printf("1.Enqueue at right\n");
        printf("2.Dequeue from left\n");
        printf("3.Dequeue from right\n");
        printf("4.Display\n");
```

```
    printf("5.Quit\n");
    printf("Enter your choice : ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      Enqueue_right();
      break;
    case 2:
      delete_left();
      break;
    case 3:
      delete_right();
      break;
    case 4:
      display_queue();
      break;
    case 5:
      break;
    default:
      printf("Wrong choice\n");
    }
  } while (choice != 5);
}
void output_que()
{
  int choice;
  do
  {
    printf("1.Enqueue at right\n");
    printf("2.Enqueue at left\n");
    printf("3.Dequeue from left\n");
    printf("4.Display\n");
    printf("5.Quit\n");
    printf("Enter your choice : ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      Enqueue_right();
      break;
    case 2:
      Enqueue_left();
      break;
    case 3:
```

```
        delete_left();
        break;
      case 4:
        display_queue();
        break;
      case 5:
        break;
      default:
        printf("Wrong choice\n");
      }
   } while (choice != 5);
}
int main()
{
   int choice;
   printf("1.Input restricted dequeue\n");
   printf("2.Output restricted dequeue\n");
   printf("Enter your choice : ");
   scanf("%d", &choice);
   switch (choice)
   {
   case 1:
      input_que();
      break;
   case 2:
      output_que();
      break;
   default:
      printf("Wrong choice\n");
   }
}
```

OUTPUT

```
INPUT - RESTRICTED DEQUEUE

1.Input restricted dequeue
2.Output restricted dequeue
Enter your choice : 1
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 1
```

```
Input the element for adding in queue : 1
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 2
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 3
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 4
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 5
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 4
Queue elements :
1 2 3 4 5
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 2
Element deleted from queue is : 1
1.Enqueue at right
```

```
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 4
Queue elements :
2 3 4 5
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 3
Element deleted from queue is : 5
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 4
Queue elements :
2 3 4
1.Enqueue at right
2.Dequeue from left
3.Dequeue from right
4.Display
5.Quit
Enter your choice : 5
--------------------------------------------------------------------------------
OUTPUT -- RESTRICTED DEQUEUE

1.Input restricted dequeue
2.Output restricted dequeue
Enter your choice : 2
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 1
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
```

```
5.Quit
Enter your choice : 2
Input the element for adding in queue : 2
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 3
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 2
Input the element for adding in queue : 4
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 5
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 4
Queue elements :
4 2 1 3 5
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 3
Element deleted from queue is : 4
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 4
```

```
Queue elements :
2 1 3 5
1.Enqueue at right
2.Enqueue at left
3.Dequeue from left
4.Display
5.Quit
Enter your choice : 5
```

Q4 Write a menu driven program to implement circular queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty using linked list.

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

struct node
{
   int info;
   struct node *link;
} *rear = NULL;

void insert(int item);
int del();
void display();
int isEmpty();
int peek();
int main()
{
   int choice, item;
   while (1)
   {
      printf("\n 1 - Enque");
      printf("\n 2 - Deque");
      printf("\n 3 - Peek");
      printf("\n 4 - Display");
      printf("\n 5 - Exit");
      printf("\nEnter your choice : ");
      scanf("%d", &choice);

      switch (choice)
      {
      case 1:
         printf("\nEnter the element for insertion : ");
         scanf("%d", &item);
         insert(item);
         break;
```

```
      case 2:
         printf("\nDeleted element is %d\n", del());
         break;
      case 3:
         printf("\nItem at the front of queue is %d\n", peek());
         break;
      case 4:
         display();
         break;
      case 5:
         exit(1);
      default:
         printf("\nWrong choice\n");
      }
   }
}
void insert(int item)
{
   struct node *tmp;
   tmp = (struct node *)malloc(sizeof(struct node));
   tmp->info = item;
   if (tmp == NULL)
   {
      printf("\nMemory not available\n");
      return;
   }

   if (isEmpty())
   {
      rear = tmp;
      tmp->link = rear;
   }
   else
   {
      tmp->link = rear->link;
      rear->link = tmp;
      rear = tmp;
   }
}

del()
{
   int item;
   struct node *tmp;
   if (isEmpty())
```

```
    {
        printf("\nQueue underflow\n");
        exit(1);
    }
    if (rear->link == rear)
    {
        tmp = rear;
        rear = NULL;
    }
    else
    {
        tmp = rear->link;
        rear->link = rear->link->link;
    }
    item = tmp->info;
    free(tmp);
    return item;
}

int peek()
{
    if (isEmpty())
    {
        printf("\nQueue underflow\n");
        exit(1);
    }
    return rear->link->info;
}
int isEmpty()
{
    if (rear == NULL)
        return 1;
    else
        return 0;
}
void display()
{
    struct node *p;
    if (isEmpty())
    {
        printf("\nQueue is empty\n");
        return;
    }
    printf("\nQueue is :\n");
    p = rear->link;
```

```
  do
  {
    printf("%d ", p->info);
    p = p->link;
  } while (p != rear->link);
  printf("\n");
}
```

OUTPUT

```
 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Enter the element for insertion : 1

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Enter the element for insertion : 2

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Enter the element for insertion : 2

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Enter the element for insertion : 4
```

```
 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 1

Enter the element for insertion : 5

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 3

Item at the front of queue is 1

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 4

Queue is :
1 2 2 4 5

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 2

Deleted element is 1

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 4

Queue is :
```

```
2 2 4 5

 1 - Enque
 2 - Deque
 3 - Peek
 4 - Display
 5 - Exit
Enter your choice : 5
```