C function of [illegible]

WRITE A PROGRAM TO IMPLEMENT A LINKED QUEUE.

```
# include <stdio.h>
# include <conio.h>
# include <malloc.h>
struct node
  {
   int data;
   struct node *next;
  };
 struct queue
   {
    struct mode *front;
    struct node * rear;
   };
 struct queme *q;
  void create_queue (struct queue *);
  struct queue *insest (struct queue *, int);
  struct queue * delete-element (struct queue *);
  struct queue * display ((struct queue*);
  int peek (struct queue *);
  int main()
   {
    int val, option;
    create_queue (q);
    clrscr();
   do
   {
    printf ("\n *** MAIN MENU ***");
    printf ("\n 1. INSERT");
    printf ("\n 2. DELETE");
    printf (" \n 3. PEEK");
    printf(" \n 4.DISPLAY);
    printf("\n 5. EXIT");
printf("\n Enter your option");
scanf("%d", &option);
(switch(option)
 {
  case 1:
        printf("\n Enter the number to insert in the Queue;");
        scanf ("%d", &val);
         q= insert (q, val);
         break;
  Case 2:
       q= delete-element(q);
       break;
  Case 3:
       val= peek(q);
       if (val != -1)
            printf ("\n The value at front of Queue is : %d", val);
            break;
  Case 4:
        q= display (q);
        break;
     }
 }
 while (option != 5);
  getch();
   return 0;
}
```

C function [illegible]

```
void create_queue(struct queue *q)
    {
        q -> rear = NULL;
        q -> front = NULL;
    }

struct queue *insert (struct queue *q, int val)
  {
    struct node * ptr;
    ptr = (struct node *) malloc (sizeof (struct node));
    ptr -> data = val;
    if (q -> front == NULL)
      {
        q -> front = ptr;
        q -> rear = ptr;
        q -> front -> next = q -> rear -> next = NULL;
      }
    else
      {
        q -> rear -> next = ptr;
        q -> rear = ptr;
        q -> rear -> next = NULL;
      }
      return q;
  }

struct queue * display (struct queue *q)
  {
    struct node *ptr;
      ptr = q -> front;
      if (ptr == NULL)
            printf ("\n Queue is empty");

      else
       {
        printf ("\n");
        while (ptr != q -> rear)
          {
           printf( "%d \t", ptr -> data);
            ptr = ptr -> next;
          }
        printf("%d \t", ptr->data);
    }
    return q;
}
struct queue * delete_element (struct queue * q)
{
    struct node * ptr;
    ptr = q->front
    if (q->front == NULL)
        printf("\n UNDERFLOW");
    else
    {
        q->front = q->front->next;
        printf("\n The value being deleted is : %d", ptr->data);
        free(ptr);
    }
    return q;
}
int peek (struct queue *q)
{
    if (q->front == NULL)
    {
        printf("\n Queue is empty");
        return -1;
    }
    else
        return q->front->data;
}
```

C function of C...

Queues Implementation with the help of array.

A queue is a FIFO (First In, First Out) data structure in which the element that is inserted first one to be taken out.
The element in a queue are added at one end called the REAR and removed from the other end called the FRONT.

Queues can be implemented by using either arrays or linked lists.

Array Representation of Queues.

→ Operation on Queues.
FRONT = 0 and REAR = 5.

12	9	7	18	14	36	
0	1	2	3	4	5	

Add → 45 @ REAR

12	9	7	18	14	36	45			
0	1	2	3	4	5	6			

Queue after insertion of New element

	9	7	18	14	36	45			
0	1	2	3	4	5	6	7	8	9

Queue after deletion of an element.
here, FRONT = 1 and REAR = 6.

ALGO TO INSERT AN ELEMENT IN A QUEUE.

```
Step1 : IF REAR = MAX-1
            WRITE "OVERFLOW"
            GOTO STEP4

Step2: IF FRONT = -1 and REAR = -1
            SET FRONT = REAR = 0

        ELSE
            SET REAR = REAR+1

        [END OF IF]

Step 3: SET QUEUE [REAR] = NUM

Step 4: EXIT.
```

ALGORITHM TO DELETE AN ELEMENT FROM A QUEUE

```
Step1: IF FRONT = -1 OR FRONT > REAR
           Write UNDERFLOW
       ELSE
           SET VAL = QUEUE[FRONT]
           SET FRONT = FRONT + 1
       [END OF IF]
Step2: EXIT
```

```
# include <stdio.h>
# include <conio.h>
# define MAX 10;
int queue[MAX];
int front = -1, rear = -1;
void insert (void);
int delete_element (void);
int peek (void);
void display (void);
int main()
{
      int option, val;
      do
      {
        printf("\n\n *** MAIN MENU ***");
        printf("\n 1. Insert an element");
        printf("\n 2. Delete an element");
        printf("\n 3. Peek");
        printf("\n 4. Display the queue");
        printf("\n 5. Exit");
        printf (" Enter your option"); ~~printf~~ scanf("%d", &option);
        switch(option)
        {
```

C function [illegible]

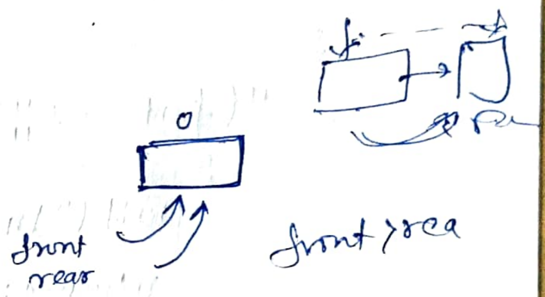
```
    case 1:
           insert();
           break;

     case 2:
         val= delete-element ();
         if(val!=-1)
         printf("\n The number is deleted is : %d", val);
         break;

      case 3:
          val= peek();
          if(val!=-1)
          printf("\n The first value in queue is : %d",val);
          break;

       case 4:
             display();
             break;
        }
    }
    while (option!=5)
     getch();
     return 0;
  }

void insert()
 {
   int num;
   printf("\n Enter the number to be inserted in the queue:");
   scanf("%d", &num);
   if (rear== MAX-1)
    printf("\n OVERFLOW");
   elseif (front==-1 && rear==-1)
   front = rear= 0;
   else
    rear++;
    queue[rear] =num;
 }
```



a[rear]
a[i]
a[i+1]=num

```
int  delete_element()
  {
    int val;
    if (front==-1 || front>rear)
      {
        printf("\n UNDERFLOW");
         return -1;
      }
     else
      {
        val= queue[front];
          front++;
         if (front> rear)
           front= rear=-1;
           return val;
      }
}

int peek()
  {
     if(front==-1 || front>rear)
       {
         printf("\n QUEUE IS EMPTY");
          return-1;
       }
      else
       {
          return queue[front];
       }
  }

  void display()
    { int i;
    printf("\n");
    If (front==-1 || front>rear)
     printf("\n Queue is Empty");
    else
    { for(i=front; i<=rear; i++) → printf("\t %d", queue[i]);
```

CIRCULAR ARRAY

INSERTING A ELEMENT IN A CIRCULAR QUEUE

```
Step1: IF FRONT=0 and REAR=MAX-1
            WRITE "OVERFLOW"
              GOTO STEP 4
            [END OF IF]

Step2: IF  FRONT=-1 and REAR=-1
              SET FRONT= REAR=0
          ELSE IF  REAR=MAX-1 AND FRONT!=0
                  SET REAR=0
          ELSE
                 SET REAR=REAR+1
           [END OF IF]

Step3: SET QUEUE [REAR]=VAL
Step4: EXIT
```

ALGORITHM TO DELETE.

```
Step1: IF FRONT=-1
            WRITE UNDERFLOW
              GOTO STEP 4
            [END OF IF]

Step2: SET VAL= QUEUE[FRONT]
Step3: IF FRONT = REAR
            SET FRONT=REAR=-1
       ELSE
          IF FRONT =MAX-1
                SET FRONT=0
          ELSE
                SET FRONT= FRONT+1
             [EOI]
Step 4! EXIT
```

C function of Circular linked list

CIRCULAR QUEUE.

```
# include <stdio.h>
# include <conio.h>
# define MAX 10
int queue[MAX];
int front=-1, rear=-1;
void insest (void);
int delele (void);
int peek (void);
void display (void);
void Insert( )
{
 int num;
 printf("\n Enter the number to be Inseoted in the queue:");
 scanf ("%d",&num);
  if (front==0 && rear= MAX-1)
        printf ("\n overflow");
  else if (front==-1 && rear==-1)
     {
      front= rear= 0;
       queue[rear]=num;
     }
    else if (rear==MAX-1 && front!=0)
       {
        rear=0;
        queue[rear]=num;
       }
     else
      {
        rear++;
        queue [rear]= num;
      }
 }
```

```
int main()
 {
  int option,val;
   clrscr();
 do
  {

  }
```

```
int delete ( )
  {
    int val;
    if( front==-1 && rear==-1)
      {
        printf ("\n underflow");
         return -1;
      }
      val = queue [front];
       if ( front==rear)
               front = rear=-1;

        else
          {  if ( front ==MAX-1)
                    front=0;

             else
                    front++;
           }
           return val;
         }


void display()
 { int i;
   printf("\n");
   if( front==-1 && rear==-1)
           printf("\n QUEUE IS EMPTY");

     else
       { if (front <rear)
           {
             for( i=front ; i<=rear ; i++)
                   printf("\t %d", queue[i]);
           }
          else
           {
```

C function of Circular linked list

```
        for ( i=front ; i<MAX ; i++)
              printf ("\t %d" , queue[i]);
        for ( i=0 ; i<=rear ; i++)
              printf ("\t %d", queue [i]);
          }
    }


int peek
    {
     if ( front ==-1 && rear ==-1)
          {
           printf ("\n Queue Is EMPTY");
           return-1;
          }
         else
          {
           return  queue[front];
          }
```