WEEK-6:

a)Write a program that implement Queue(its operations)Using Arrays

Program:

```
#include <stdio.h>
#define MAX 50
void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
int main()

{

   int choice;

   while (1)

   {

      printf("1.Insert element to queue \n");
```

```
printf("2.Delete element from queue \n");

printf("3.Display all elements of queue \n");

printf("4.Quit \n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

   case 1: insert();

   break;

   case 2: delete();

   break;
```

```
            case 3:  display();

            break;

            case 4:  exit(1) ;

            default:

            printf("Wrong choice \n");

        } /* End of switch */

    } /* End of while */

return 0;
} /* End of main() */



void insert()
```

```
{
    int add_item;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty */
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
```

```
        rear = rear + 1;

        queue_array[rear] = add_item;

    }

} /* End of insert() */



void delete()

{

    if (front == - 1 || front > rear)

    {

        printf("Queue Underflow \n");

        return ;
```

```
    }

    else

    {

        printf("Element deleted from queue is : %d\n",
queue_array[front]);

        front = front + 1;

    }

} /* End of delete() */




void display()

{

    int i;
```

```
    if (front == - 1)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
        printf("\n");
    }
} /* End of display() */
```

b) Write a program that implement Queue(its operations)Using Linked list

Program:

```
#include <stdio.h>

#include <stdlib.h>

struct node

{
```

```
    int info;
    struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

int main()
{
    int no, ch, e;

    printf("\n 1 - Enque");
    printf("\n 2 - Deque");
    printf("\n 3 - Front element");
    printf("\n 4 - Empty");
```

```
printf("\n 5 - Exit");
printf("\n 6 - Display");
printf("\n 7 - Queue size");
create();
while (1)
{
   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 = frontelement();
      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:
            exit(0);
         case 6:
            display();
            break;
         case 7:
            queuesize();
            break;
         default:
            printf("Wrong choice, Please enter correct choice  ");
            break;
         }
      }
return 0;
}
```

```
/* Create an empty queue */
void create()
{
   front = rear = NULL;
}

/* Returns queue size */
void queuesize()
{
   printf("\n Queue size : %d", count);
}

/* Enqueing the queue */
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;
    }
    count++;
}

/* Displaying the queue elements */
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);
}

/* Dequeing the queue */
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;
        }
        count--;
}

/* Returns the front element of queue */
int frontelement()
{
    if ((front != NULL) && (rear != NULL))
        return(front->info);
    else
        return 0;
```

```
}
/* Display if queue is empty or not */
void empty()
{
    if ((front == NULL) && (rear == NULL))
       printf("\n Queue empty");
   else
      printf("Queue not empty");
}
```