Week 7:

Write a C Program to implement a Double Ended Queue ADT using

i) Arrays

Program:

```
#include<stdio.h>
#define size 5
int a[size],front=-1,rear=-1,count=0;
void add_at_beg( int item)
{
        int i;
        if(front==-1)
        {
                front++;
                rear++;
                a[rear]=item;
                count++;
        }
        else if(rear>=size-1)
        {
                printf("overflow \n");
                return;
        }
         else
```

```
                {
                        for(i=count;i>0;i--)
                        {
                                a[i]=a[i-1];
                        }
                          a[i]=item;
                          count++;
                          rear++;
                 }
}
void add_at_end(int item)
{
                  if(front==-1)
                  {
                          front++;
                          rear++;
                          a[rear]=item;
                          count++;
                    }
                 else if(rear>=size-1)
                {
                          printf("overflow \n");
                          return;
```

```
            }
            else
            {
                    a[++rear]=item;
            }
 }
  void display()
 {
              int i;
             for(i=front;i<=rear;i++)
             {
                      printf("%d  ",a[i]);
             }
}
void del_fr_front()
{
              if(front==-1)
              {
                      printf("deletion impossible \n");
                      return;
              }
             else
             {
```

```
                    if(front==rear)
                    {
                            front=rear=-1;
                            return;
                    }
                   printf("the deleted element is %d \n",a[front]);
                    front=front+1;
                    count--;
                }
}
void del_fr_rear()
{
                if(front==-1)
                 {
                        printf("deletion is impossible \n");
                        return;
                 }
                else
                {
                        if(front==rear)
                        {
                                front=rear=-1;
                        }
```

```
                printf("Deleted element is  %d \n",a[rear]);
                rear=rear-1;
                count--;
            }
}
int main()
{
        int ch,element;
        while(1)
        {
            printf("\n Dequeue Menu \n");
            printf("\n 1-add at beg \n");
            printf("\n 2-add at end \n");
            printf("\n 3-del from front \n");
            printf("\n 4-del from rear \n");
            printf("\n 5-display \n");
            printf("\n 6-exit \n");
            printf("enter your choice \n");
            scanf("%d",&ch);
            switch(ch)
            {
                case 1:printf("enter element \n");
                    scanf("%d",&element);
```

```
                    add_at_beg(element);
                    break;
                case 2:printf("enter element \n");
                    scanf("%d",&element);
                    add_at_end(element);
                    break;
                case 3:del_fr_front();
                    break;
                case 4:del_fr_rear();
                    break;
                case 5:display();
                    break;
                case 6:exit(0);
                    break;
                default:printf("invalid statement \n");
            }
        }
 return 0;
}
```

Week 7:

Write a C Program to implement a Double Ended Queue ADT using

ii) Double linked list

Program:

```
#include <stdio.h>
#include<stdlib.h>
  struct node
 {
            int data;
            struct node *prev, *next;
  };
 struct node *front = NULL, *rear = NULL;
 /* insertion at the front of the queue */
  void enqueueAtFront(int data)
 {
            struct node *newnode, *temp;
            newnode = (struct node *)malloc(sizeof (struct node));
            newnode->data = data;
             if(front==NULL && rear==NULL)
            {
                        newnode->next = newnode->prev = NULL;
                        front=rear=newnode;
             }
             else
```

```
                {
                    temp=front;
                    temp->prev=newnode;
                    newnode->next=temp;
                    newnode->prev=NULL;
                    front=newnode;
                }
}
 /*insertion at the rear of the queue */
  void enqueueAtRear(int data)
 {
                  struct node *newnode, *temp;
                newnode =(struct node *)malloc(sizeof (struct node));
                newnode->data = data;
                if(front==NULL && rear==NULL)
                {
                    newnode->next = newnode->prev = NULL;
                    front=rear=newnode;
                }
                else
                {
```

```
                temp=rear;
                temp->next=newnode;
                newnode->next=NULL;
                newnode->prev=temp;
                rear=newnode;
        }
}
 /* deletion at the front of the queue */
 void dequeueAtFront()
{
        struct node *temp;
        if(front==NULL&&rear==NULL)
        {
                printf("dequeue is empty\n");
                return;
        }
         else if(front==rear)
        {
                temp=front;
                free(temp);
                front=rear=NULL;
```

```
        }
    else
    {
        temp=front;
        front=temp->next;
        free(temp);
    }
    return;
}
/* deletion at the rear of the queue */
void dequeueAtRear()
{
    struct node *temp;
    if(front==NULL&&rear==NULL)
    {
        printf("dequeue is empty\n");
        return;
    }
     else if(front==rear)
    {
        temp=front;
```

```
                free(temp);
                front=rear=NULL;
        }
        else
        {
                temp=rear;
                rear=temp->prev;
                free(temp);
        }
        return;
  }
/* display elements present in the queue */
  void display()
{
        struct node *temp;
        if(front==NULL&&rear==NULL)
        {
                printf("no element to display\n");
                return;
        }
        else if(front==rear)
```

```
        {
            temp=front;
            printf("%d",temp->data);
            return;
        }
        else
        {
              for(temp=front;temp!=rear;temp=temp->next)
              {
                    printf("%d ",temp->data);
              }
            printf("%d  ",rear->data);;
            printf("\n");
        }
}
int main()
{
        int data, ch;
        while (1)
         {
```

```
                printf("1. Enqueue at front\n2. Enqueue at
rear\n");
                printf("3. Dequeue at front\n4. Dequeue at
rear\n");
                printf("5. Display\n6. Exit\n");
                printf("Enter your choice:");
                scanf("%d", &ch);
                switch (ch)
               {
                      case 1:
                            printf("Enter the data to insert:");
                            scanf("%d", &data);
                            enqueueAtFront(data);
                            break;
                     case 2:
                            printf("Enter ur data to insert:");
                            scanf("%d", &data);
                            enqueueAtRear(data);
                            break;
                    case 3:
                            dequeueAtFront();
```

```
                    break;
                case 4:
                    dequeueAtRear();
                    break;
                case 5:
                    display();
                    break;
                case 6:
                    exit(0);
                default:
                    printf("Pls. enter correct option\n");
                    break;
            }
    }
            return 0;
    }
```