**#include <stdio.h>**

**#include <stdlib.h>**

**struct node {**

**int data;**

**struct node \*prev, \*next;**

**};**

**struct node \*head = NULL, \*tail = NULL;**

**struct node \* createNode(int data) {**

**struct node \*newnode = (struct node \*)malloc(sizeof (struct node));**

**newnode->data = data;**

**newnode->next = newnode->prev = NULL;**

**return (newnode);**

**}**

**/\***

**\* create sentinel(dummy head & tail) that**

**\* helps us to do insertion and deletion**

**\* operation at front and rear so easily. And**

**\* these dummy head and tail wont get deleted**

**\* till the end of execution of this program**

**\*/**

**void createSentinels() {**

**head = createNode(0);**

**tail = createNode(0);**

**head->next = tail;**

**tail->prev = head;**

**}**

**/\* insertion at the front of the queue \*/**

**void enqueueAtFront(int data) {**

**struct node \*newnode, \*temp;**

**newnode = createNode(data);**

**temp = head->next;**

**head->next = newnode;**

**newnode->prev = head;**

**newnode->next = temp;**

**temp->prev = newnode;**

**}**

**/\*insertion at the rear of the queue \*/**

**void enqueueAtRear(int data) {**

**struct node \*newnode, \*temp;**

**newnode = createNode(data);**

**temp = tail->prev;**

**tail->prev = newnode;**

**newnode->next = tail;**

**newnode->prev = temp;**

**temp->next = newnode;**

**}**

**/\* deletion at the front of the queue \*/**

**void dequeueAtFront() {**

**struct node \*temp;**

**if (head->next == tail) {**

**printf("Queue is empty\n");**

**} else {**

**temp = head->next;**

**head->next = temp->next;**

**temp->next->prev = head;**

**free(temp);**

**}**

**return;**

**}**

**/\* deletion at the rear of the queue \*/**

**void dequeueAtRear() {**

**struct node \*temp;**

**if (tail->prev == head) {**

**printf("Queue is empty\n");**

**} else {**

**temp = tail->prev;**

**tail->prev = temp->prev;**

**temp->prev->next = tail;**

**free(temp);**

**}**

**return;**

**}**

**/\* display elements present in the queue \*/**

**void display() {**

**struct node \*temp;**

**if (head->next == tail) {**

**printf("Queue is empty\n");**

**return;**

**}**

**temp = head->next;**

**while (temp != tail) {**

**printf("%-3d", temp->data);**

**temp = temp->next;**

**}**

**printf("\n");**

**}**

**void main() {**

**int data, ch;**

**createSentinels();**

**while (1) {**

**system("cls");**

**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:**

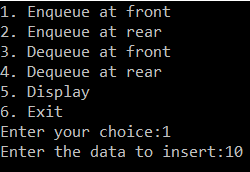
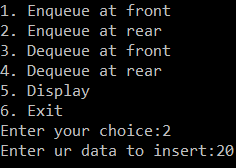
**exit(0);**

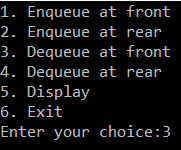
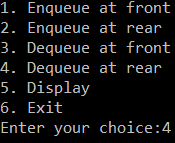
**default:**

**printf("Pls. enter correct option\n");**

**break;**

**}}}**

** **

** **

**#include<stdio.h>**

**#include<stdlib.h>**

**#define MAX 30**

**typedef struct dequeue**

**{**

**int data[MAX];**

**int rear,front;**

**}dequeue;**

**void initialize(dequeue \*p);**

**int empty(dequeue \*p);**

**int full(dequeue \*p);**

**void enqueueR(dequeue \*p,int x);**

**void enqueueF(dequeue \*p,int x);**

**int dequeueF(dequeue \*p);**

**int dequeueR(dequeue \*p);**

**void print(dequeue \*p);**

**void main()**

**{**

**int i,x,op,n;**

**dequeue q;**

**initialize(&q);**

**do**

**{**

**printf("\n1.Create\n2.Insert(rear)\n3.Insert(front)\n4.Delete(rear)\n5.Delete(front)");**

**printf("\n6.Print\n7.Exit\n\nEnter your choice:");**

**scanf("%d",&op);**

**switch(op)**

**{**

**case 1: printf("\nEnter number of elements:");**

**scanf("%d",&n);**

**initialize(&q);**

**printf("\nEnter the data:");**

**for(i=0;i<n;i++)**

**{**

**scanf("%d",&x);**

**if(full(&q))**

**{**

**printf("\nQueue is full!!");**

**exit(0);**

**}**

**enqueueR(&q,x);**

**}**

**break;**

**case 2: printf("\nEnter element to be inserted:");**

**scanf("%d",&x);**

**if(full(&q))**

**{**

**printf("\nQueue is full!!");**

**exit(0);**

**}**

**enqueueR(&q,x);**

**break;**

**case 3: printf("\nEnter the element to be inserted:");**

**scanf("%d",&x);**

**if(full(&q))**

**{**

**printf("\nQueue is full!!");**

**exit(0);**

**}**

**enqueueF(&q,x);**

**break;**

**case 4: if(empty(&q))**

**{**

**printf("\nQueue is empty!!");**

**exit(0);**

**}**

**x=dequeueR(&q);**

**printf("\nElement deleted is %d\n",x);**

**break;**

**case 5: if(empty(&q))**

**{**

**printf("\nQueue is empty!!");**

**exit(0);**

**}**

**x=dequeueF(&q);**

**printf("\nElement deleted is %d\n",x);**

**break;**

**case 6: print(&q);**

**break;**

**default: break;**

**}**

**}while(op!=7);**

**}**

**void initialize(dequeue \*P)**

**{**

**P->rear=-1;**

**P->front=-1;**

**}**

**int empty(dequeue \*P)**

**{**

**if(P->rear==-1)**

**return(1);**

**return(0);**

**}**

**int full(dequeue \*P)**

**{**

**if((P->rear+1)%MAX==P->front)**

**return(1);**

**return(0);**

**}**

**void enqueueR(dequeue \*P,int x)**

**{**

**if(empty(P))**

**{**

**P->rear=0;**

**P->front=0;**

**P->data[0]=x;**

**}**

**else**

**{**

**P->rear=(P->rear+1)%MAX;**

**P->data[P->rear]=x;**

**}**

**}**

**void enqueueF(dequeue \*P,int x)**

**{**

**if(empty(P))**

**{**

**P->rear=0;**

**P->front=0;**

**P->data[0]=x;**

**}**

**else**

**{**

**P->front=(P->front-1+MAX)%MAX;**

**P->data[P->front]=x;**

**}**

**}**

**int dequeueF(dequeue \*P)**

**{**

**int x;**

**x=P->data[P->front];**

**if(P->rear==P->front) //delete the last element**

**initialize(P);**

**else**

**P->front=(P->front+1)%MAX;**

**return(x);**

**}**

**int dequeueR(dequeue \*P)**

**{**

**int x;**

**x=P->data[P->rear];**

**if(P->rear==P->front)**

**initialize(P);**

**else**

**P->rear=(P->rear-1+MAX)%MAX;**

**return(x);**

**}**

**void print(dequeue \*P)**

**{**

**if(empty(P))**

**{**

**printf("\nQueue is empty!!");**

**exit(0);**

**}**

**int i;**

**i=P->front;**

**while(i!=P->rear)**

**{**

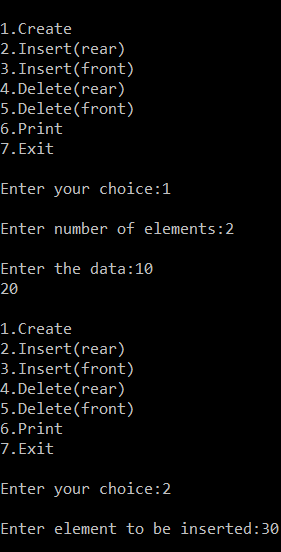
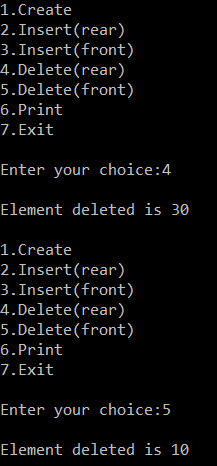
**printf("\n%d",P->data[i]);**

**i=(i+1)%MAX;**

**}**

**printf("\n%d\n",P->data[P->rear]);**

**}**

** **