



**19BIT0292**

**Bhaumik Tandan**

**LAB SHEET**

**DATA STRUCTURES**  
**AND**  
**ALGORITHMS**  
**LABORATORY**

**CSE2011**

**L57+L58**

## Q1) Implementation of double ended queue using array.

### CODE

```
#include<stdio.h>

#define s 20

int ar[s],f=0,r=-1,e;

add_rear(int a)
{
    if(e==s){
        printf("\nQueue Full");
        return 1;
    }
    r=(r+1)%s;
    ar[r]=a;
    e++;
    printf("\n%d added to rear",a);
}

add_front(int a)
{
    if(e==s){
        printf("\nQueue Full");
        return 1;
    }
    f-=1;
    if(f==-1)
        f=s-1;
    ar[f]=a;
```

```
e++;  
  
printf("\n%d added to front",a);  
}  
del_front()  
{  
    if(e==0)  
    {  
        printf("\nQueue Empty");  
        return;  
    }  
  
    e--;  
  
    int t=f;  
  
    f=(f+1)%s;  
  
    return ar[t];  
}  
del_rear()  
{  
    if(e==0)  
    {  
        printf("\nQueue Empty");  
        return;  
    }  
  
    e--;  
  
    int t=r;  
  
    r-=1;  
  
    if(r== -1)  
  
    r=s-1;  
  
    return ar[t];  
}
```

```
disp()
{
    if(e==0)
    {
        printf("\nQueue Empty");
        return;
    }
    printf("\nFront ---> ");
    int i=f;
    while (i!=r)
    {
        printf("%d ",ar[i]);
        i=(i+1)%s;
    }
    printf("%d ---> Rear",ar[i]);
}
```

```
main()
{
    add_rear(3);
    add_front(23);
    add_rear(94);
    disp();
    printf("\n %d deleted from front",del_front());
    add_rear(232);
    add_front(4);
    disp();
    printf("\n %d deleted from rear",del_rear());
}
```

# OUTPUT

```
3 added to rear
23 added to front
94 added to rear
Front ---> 23 3 94 ---> Rear
23 deleted from front
232 added to rear
4 added to front
Front ---> 4 3 94 232 ---> Rear
232 deleted from rear
```

[CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE](#)

## Q2) Implementation of doubly linked list.

### CODE

```
#include<stdio.h>

struct node
{
    int d;
    struct node *n;
    struct node *p;
} * h;

ins_r(int n)
{
    struct node *t = (struct node *)malloc(sizeof(struct node));
    t->d = n;
```

```

t->n = 0;

printf("\n%d inserted at rear",n);

if (!h){
    h = t;
    return;
}

struct node *p = h;
while (p->n)
    p = p->n;
p->n = t;
t->p=p;
}

```

```

del_r()
{
    if (!h || !(h->n))
    {
        h=0;
        return;
    }

    struct node *p = h;
    while (p->n->n)
        p = p->n;
    p->n=0;
}

```

```

ins_f(int n)
{
    struct node *t = (struct node *)malloc(sizeof(struct node));

```

```

printf("\n%d inserted at front",n);

t->d = n;

t->n = h;

t->p=0;

if(h)

h->p=t;

h=t;
}

disp()
{
    struct node *t = h;

    printf("\n");

    while (t)

    {

        printf("%d ",t->d);

        t = t->n;

    }

}

ins_p(int n,int po)
{
    struct node *t = (struct node *)malloc(sizeof(struct node));

    t->d = n;

    t->n = 0;

    printf("\n%d inserted at position %d",n,po);

    if (!h)

    {

        h = t;

        return;

    }

```

```

struct node *p = h;
while (p->n && --po>0)
    p = p->n;
p->n = t;
t->p=p;
}

```

```

del_f()
{
    if(!h)
        return;
    h=h->n;
    if(h)
        h->p=0;
}

```

```

del_p(int po)
{
    if (!h || !(h->n))
    {
        h = 0;
        return;
    }
    struct node *p = h;
    while (p->n->n && --po>1)
        p = p->n;
    struct node *t=p->n;
    p->n=t->n;
    t->n->p=p;
}

```



```
}
```

```
main()
```

```
{
```

```
    ins_r(21);
```

```
    ins_f(42);
```

```
    ins_f(4290);
```

```
    ins_r(90);
```

```
    disp();
```

```
    del_r();
```

```
    disp();
```

```
    del_f();
```

```
    disp();
```

```
    ins_f(426);
```

```
    ins_r(490);
```

```
    disp();
```

```
    del_p(2);
```

```
    disp();
```

```
}
```

## OUTPUT

```
21 inserted at rear
42 inserted at front
4290 inserted at front
90 inserted at rear
4290 42 21 90
4290 42 21
42 21
426 inserted at front
490 inserted at rear
426 42 21 490
426 21 490
```

[CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE](#)

### Q3) Realization of double ended queue with doubly linked list.

## CODE

```
#include <stdio.h>

struct node
{
    int d;
    struct node *n;
    struct node *p;
} * f, *r;

struct node *make_node(int a, struct node *n, struct node *p)
{
    struct node *t = (struct node *)malloc(sizeof(struct node));
    t->d = a;
    t->n = n;
    t->p = p;
    return t;
}

add_rear(int a)
{
    printf("\n%d inserted at the rear",a);
    if (!f)
    {
        f=r=make_node(a,0,0);
```

```

        return;
    }
    struct node *p = f;
    while (p->n)
        p = p->n;
    p->n=make_node(a,0,p);
    r=p->n;
}

```

```

add_front(int a)
{
    printf("\n%d inserted at the front",a);
    if (!f)
    {
        f=r=make_node(a,0,0);
        return;
    }

```

```

    f=make_node(a,f,0);
    f->n->p=f;
}

```

```

del_front()
{
    if(!f)
        return;
    int a=f->d;
    f=f->n;
    if(f)

```

```
f->p=0;  
return a;  
}
```

```
del_rear()  
{  
    if(!f)  
        return;  
    int a=r->d;  
    r=r->p;  
    if(r)  
        r->n=0;  
    return a;  
}
```

```
disp()  
{  
    struct node *t = f;  
    printf("\n");  
    while (t)  
    {  
        printf("%d ",t->d);  
        t = t->n;  
    }  
}
```

```
main()  
{  
    add_rear(3);
```

```

add_front(23);
add_rear(94);
disp();
printf("\n %d deleted from front",del_front());
add_front(4);
add_rear(232);
disp();
printf("\n %d deleted from rear",del_rear());
disp();
}

```

## OUTPUT

```

3 inserted at the rear
23 inserted at the front
94 inserted at the rear
23 3 94
23 deleted from front
4 inserted at the front
232 inserted at the rear
4 3 94 232
232 deleted from rear
4 3 94

```

[CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE](#)

### Q4) Perform Linear Search.

## CODE

```

#include<stdio.h>

search(int p,int *a,int s)
{
    if(p==0 || a[p-1]==s)

```

```

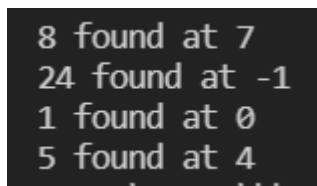
    return p-1;

    return search(p-1,a,s);
}

main()
{
    int a[]={1,2,3,4,5,6,7,8};
    printf("\n%d found at %d",8,search(8,a,8));
    printf("\n%d found at %d",24,search(8,a,24));
    printf("\n%d found at %d",1,search(8,a,1));
    printf("\n%d found at %d",5,search(8,a,5));
}

```

## OUTPUT



```

8 found at 7
24 found at -1
1 found at 0
5 found at 4

```

[CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE](#)

### Q5) Preform Binary Search.

## CODE

```

#include<stdio.h>

search(int st,int en,int *a,int s)
{

```

```

    if(st+1>=en)
    return -1;
    int m=st+(en-st)/2;
    if(a[m]==s)
    return m;
    if(a[m]>s)
    return search(st,m,a,s);
    if(a[m]<s)
    return search(m,en,a,s);
}

main()
{
    int a[]={1,2,3,4,5,6,7,8,10,11,12};
    printf("\n%d found at %d",4,search(0,12,a,4));
    printf("\n%d found at %d",5,search(0,12,a,5));
    printf("\n%d found at %d",10,search(0,12,a,10));
    printf("\n%d found at %d",-32,search(0,12,a,-32));
    printf("\n%d found at %d",832,search(0,12,a,832));
}

```

## OUTPUT

```

4 found at 3
5 found at 4
10 found at 8
-32 found at -1
832 found at -1

```

[CLICK HERE FOR GITHUB LINK OF WHOLE SOURCE CODE](#)