DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Year: 2nd **Semester: 3rd**

Data Structures- PCS-303

LAB MANUAL

Prepared By: HOD(CSE)

<Ms.Ritu Rawat><Mr.Dhajvir Rai>

DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

INDEX

S.No	Practical's Name	Date	Remark
1.	Write a C program to find the sum of all elements in an array which are at odd position		
2.	Write a C program to reverse the array's elements.		
3.	a. Write a C program to search for a number in an array. b. Write a C program to merge two sorted array into third sorted array		
4.	Write a C program to implement a stack using array with the following operations: Push (), Pop(), Display().		
5.	Write a C program to implement a linear queue using array with the following operations: Enqueue(), Dequeue(), Display().		
6.	Write a C program to create, insert and delete a node from a singly linked list. (consider all the cases)		
7.	Write a C program to reverse a singly linked list.		
8.	Write a C program to perform bubble, selection and insertion sort.		
9.	Write a C program to perform merge and quick sort.		

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :Data Structures Lab	**EXPERIMENT NO. 1**	
	Course Code : PCS 302 **Faculty :Ms. Ritu Rawat**	**Branch: CSE**	**Semester:III**

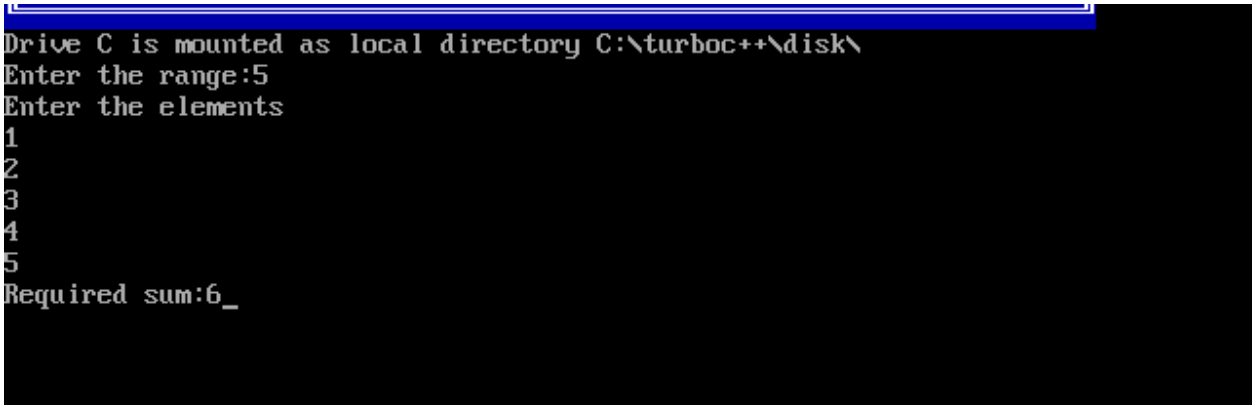
OBJECTIVE: Write a program to find the sum of all elements in an array which are at odd position.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int sum(int [],int);
int main()
{
int a[20],n,i;
printf("Enter the range:");
scanf("%d",&n);
printf("Enter the elements\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Required sum:%d",sum(a,n));
getch();
return 0;
}
int sum(int a[],int n)
{
int i,s=0;
for(i=0;i<n;i++)
```

```
{
if(i%2!=0)
s=s+a[i];
}
return s;
}
```

OUTPUT:



OUTCOME:

1) To program to find the sum of all elements in an array which are at odd position

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :Data Structures Lab	**EXPERIMENT NO. 2**	
	Course Code : PCS 302 **Faculty :Ms. Ritu Rawat**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program to reverse the array's elements.

```
#include<stdio.h>
#include<conio.h>
void reverse(int [],int);
int main()
{
        int a[20],n,i;
        printf("Enter the range:");
        scanf("%d",&n);
        printf("Enter the elements\n");
        for(i=0;i<n;i++)

                scanf("%d",&a[i]);
  reverse(a,n);
  printf("Reversed array\n");
        for(i=0;i<n;i++)
                printf("%d ",a[i]);
        getch();
  return 0;
}
void reverse(int a[],int n)
```

```
{
    int i,j,t;
    for(i=0,j=n-1;i<j;i++,j--)
    {

        t=a[i];
        a[i]=a[j];
        a[j]=t;
    }

}
```

OUTPUT:

OUTCOME:

1) To program to reverse the array's elements

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :Data Structures Lab	**EXPERIMENT NO. 4**	
	Course Code : PCS 302 **Faculty : Ms. Ritu Rawat**	**Branch: CSE**	**Semester:III**

OBJECTIVE:a. Write a program to search for a number in an array

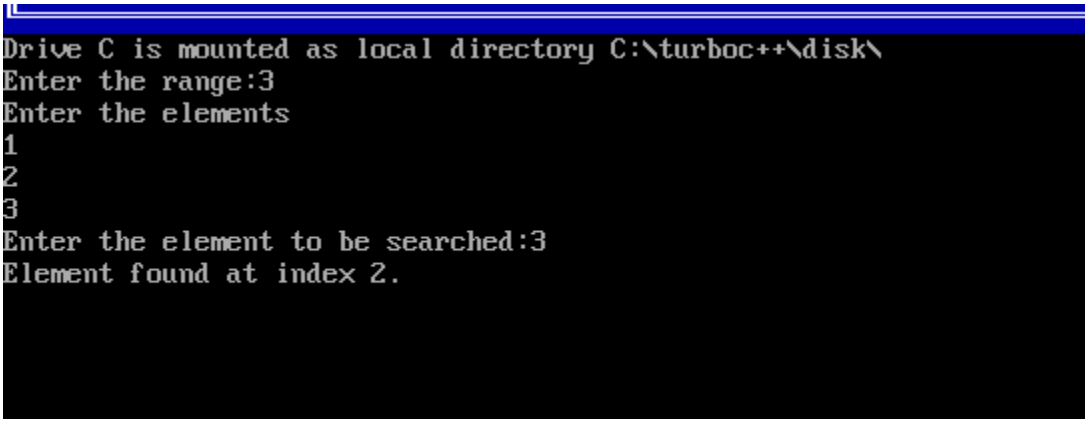
b. Write a program to merge two sorted array into third sorted array.

PROGRAM: a)

```
#include<stdio.h>
#include<conio.h>
void search(int [],int);
int main()
{
        int a[20],n,i;
        printf("Enter the range:");
        scanf("%d",&n);
        printf("Enter the elements\n");
        for(i=0;i<n;i++)

                scanf("%d",&a[i]);
  search(a,n);
        getch();
  return 0;
}
void search(int a[],int n)
{
```

```
int i,k,f=-1;
printf("Enter the element to be searched:");
scanf("%d",&k);
for(i=0;i<n;i++)
{
        if(k==a[i])
        {
          f=i;
          break;
   }
}
if(f==-1)
   printf("Element not found!");
else
   printf("Element found at index %d.",f);
   }
```

OUTPUT:



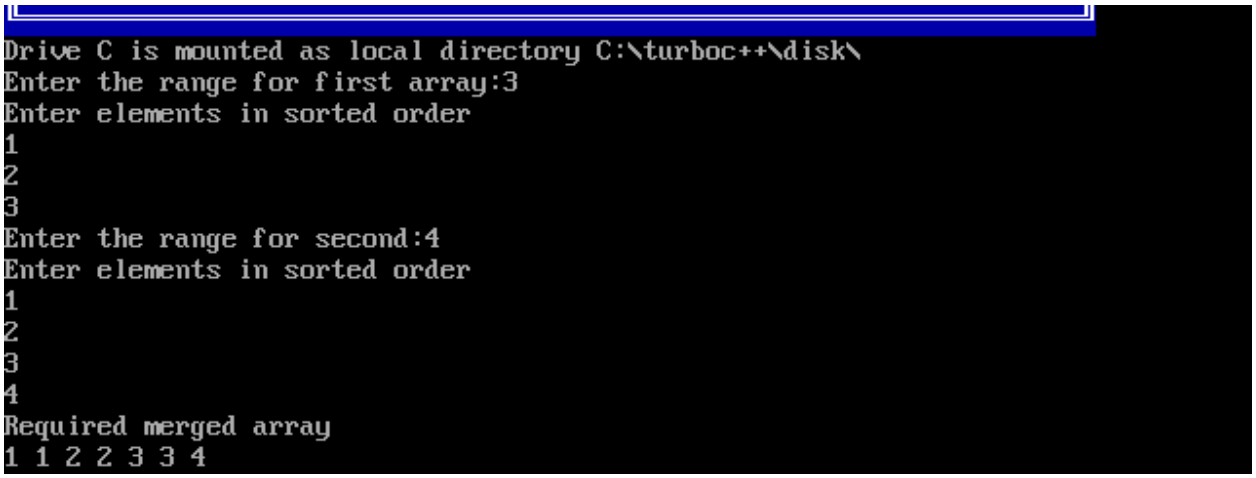
b.. Write a program to merge two sorted array into third sorted array.

```
#include<stdio.h>
#include<conio.h>
void merge(int[],int[],int[],int,int);
int main()
{
int a[50],b[50],c[50],m,n,i;
printf("Enter the range for first array:");
scanf("%d",&m);
printf("Enter elements in sorted order\n");
for(i=0;i<m;i++)

scanf("%d",&a[i]);
printf("Enter the range for second:");
scanf("%d",&n);
```

```
printf("Enter elements in sorted order\n");
for(i=0;i<n;i++)
scanf("%d",&b[i]);
merge(a,b,c,m,n);
printf("Required merged array\n");
for(i=0;i<(m+n);i++)
printf("%d ",c[i]);
getch();
return 0;
}
void merge(int a[],int b[],int c[],int m,int n)
{
int i,j,k;
i=j=k=0;
while(i<m&&j<n)
{
if(a[i]<b[j])
{
c[k]=a[i];
k++,i++;
}
else
{
c[k]=b[j];
k++,j++;
```

```
}

}

while(i<m)

{

c[k]=a[i];

k++,i++;

}

while(j<n)

{


c[k]=b[j];

k++,j++;

}


}
```

OUTPUT:



OUTCOME:

1) To Newton's Forward and Backward Interpolation formula using C programming.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :Data Structures Lab	**EXPERIMENT NO. 4**	
	Course Code : PCS 302 **Faculty :Ms. Ritu Rawat**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program to implement a linear queue using array with the following operations:

Enqueue (), Dequeue (), Display ()

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define max 5
void enque(int);
int deque();
void display();
int queue[max];

int f=-1,r=-1;
int main()
{
int ch,n;
while(1)
{
printf("1. Enqueue\n");
printf("2. Dequeue\n");
```

```
printf("3. Display\n");
printf("4. Exit\n");
printf("Enter choice:");
scanf("%d",&ch);

switch(ch)
{
case 1:printf("Enter a number:");
scanf("%d",&n);
enque(n);
break;
case 2:ch=deque();
if(ch==-1)
printf("Queue is empty!\n");
else
printf("Deleted Element:%d\n",ch);
break;
case 3:display();
break;

case 4:return 0;
default:  printf("Wrong choice\n");
}
}
return 0;

}
```

```
void enque(int item)
{
if(r==max-1)
{
printf("Queue is full!\n");
}
else if(r==-1)
f=r=0,queue[r]=item;
else
{
r++;
queue[r]=item;
}

}

int deque()
{
int t;
if(f==-1)
{
return -1;
}
else if(f==r)
{
t=queue[f];
```

```
f=r=-1;
}
else
{
t=queue[f];
f++;
}
return t;
}

void display()
{
int i;

printf("Queue elements\n");
for(i=f;i<=r;i++)
printf("%d\n",queue[i]);

}
```

OUTPUT:

```
Drive C is mounted as local directory C:\turboc++\disk\
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice:1
Enter a number:12
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice:2
Deleted Element:12
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice:23
Wrong choice
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice:4
```

OUTCOME:

To implementEnqueue (), Dequeue (), Display ()

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :Data Structures Lab	**EXPERIMENT NO. 5**	
	Course Code : PCS 302 **Faculty :Ms. Ritu Rawat**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program to implement a linear queue using array with the following operations:

Enqueue (), Dequeue (), Display ()

Program:

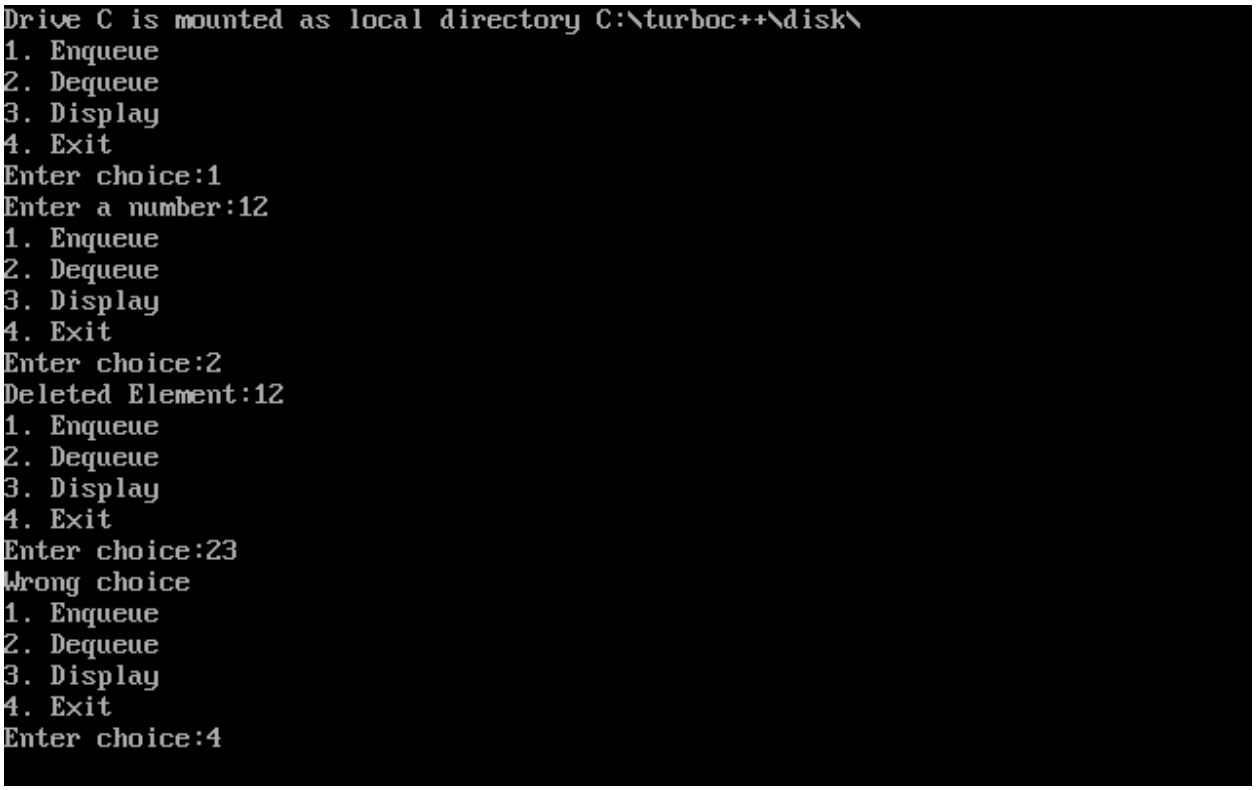
```
#include<stdio.h>

#include<conio.h>

#define max 5

void enque(int);

int deque();

void display();

int queue[max];


int f=-1,r=-1;

int main()

{

int ch,n;

while(1)

{

printf("1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");
```

```
printf("4. Exit\n");
printf("Enter choice:");
scanf("%d",&ch);

switch(ch)
{
case 1:printf("Enter a number:");
scanf("%d",&n);
enque(n);
break;
case 2:ch=deque();
if(ch==-1)
printf("Queue is empty!\n");
else
printf("Deleted Element:%d\n",ch);
break;
case 3:display();
break;

case 4:return 0;
default:  printf("Wrong choice\n");
}
}
return 0;

}
```

```
void enque(int item)
{
if(r==max-1)
{
printf("Queue is full!\n");
}
else if(r==-1)
f=r=0,queue[r]=item;
else
{
r++;
queue[r]=item;
}

}

int deque()
{
int t;
if(f==-1)
{
return -1;
}
else if(f==r)
{
t=queue[f];
```

```
f=r=-1;
}
else
{
t=queue[f];
f++;
}
return t;
}

void display()
{
int i;

printf("Queue elements\n");
for(i=f;i<=r;i++)
printf("%d\n",queue[i]);

}
```

OUTPUT:



OUTCOME:

To implementEnqueue (), Dequeue (), Display ()

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name :Data Structures Lab	**EXPERIMENT NO. 6**	
	Course Code : PCS 302 **Faculty :Ms. Ritu Rawat**	**Branch: CSE**	**Semester:III**

OBJECTIVE: Write a program to create, insert and delete a node from a singly linked list. (consider all the cases)

Program:

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

typedef struct nd

{

   int info;

   struct nd *next;

}node;

node *insert(node *);

node *insert_beg(node *);


node *insert_end(node *);

node *insert_btw(node *);

node *del(node *head);

node *del_beg(node *head);

node *del_end(node *head);
```

```
node *del_btw(node *head);
int count_nodes(node *);
void display(node *);
int main()
{
        node *head=NULL;
        int ch,n;

        while(1)
        {
                printf("1. Insert\n");
                printf("2. Delete\n");
                printf("3. Display\n");
                printf("4. Exit\n");
                printf("Enter choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:     head=insert(head);
                                                break;
                        case 2:    head=del(head);

                break;
                        case 3:    display(head);
                                    break;
                        case 4:    return 0;
                        default:  printf("Wrong choice\n");
```

```
                    }
                }
                getch();
}
i=0;
do {
i++;
}
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*(diff[i][1]+diff[i-1][1])/2;
y2=p*p*diff[i-1][2]/2;
y3=p*(p*p-1)*(diff[i-1][3]+diff[i-2][3])/6;
y4=p*p*(p*p-1)*diff[i-2][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\n\n When x=%6.2f, y=%6.8f",x,y);
getch();
}
        node *insert(node *head)
        {
            int ch;
            printf("1.Insert in begin\n");
            printf("2.Insert in end\n");
            printf("3.Insert in between\n");
            printf("4.Roll back to menu\n");
            printf("Enter choice:");
```

```
        scanf("%d",&ch);
        switch(ch)
        {
                case 1:         head=insert_beg(head);
                                        break;
                case 2:         head=insert_end(head);
                                        break;
                case 3:         head=insert_btw(head);
                                        break;
                case 4:         break;
                default: printf("Wrong choice!\n");
        }
        return head;
}

node *insert_beg(node *head)
{
        node *p;
        p=(node*)malloc(sizeof(node));
        printf("Enter value for node:");
        scanf("%d",&p->info);
        p->next=NULL;
        if(head==NULL)
        {
                head=p;
```

```
        }
        else
        {
                p->next=head;
                head=p;
        }
        return head;
}
    node *insert_end(node *head)
    {
        node *p,*t;
        p=(node*)malloc(sizeof(node));
        printf("Enter value for node:");
        scanf("%d",&p->info);
        p->next=NULL;
        if(head==NULL)
        {
                head=p;
        }
        else
        {
                t=head;
                while(t->next!=NULL)
                {
```

```
                t=t->next;
            }
            t->next=p;
        }
        return head;
}

int count_nodes(node *head)

{
        node *t;
        int c=0;
        t=head;
        while(t!=NULL)
        {
            c++;
            t=t->next;
        }
        return c;
}
  node *insert_btw(node *head)
  {
      node *p,*t;
      int cn,pos,i;
      p=(node*)malloc(sizeof(node));

      printf("Enter the position of the node to be inserted:");
```

```
scanf("%d",&pos);
cn=count_nodes(head);
if(pos>0&&pos<cn+2)
{
        if(pos==1)
        {
                head=insert_beg(head);
        }
        else if(pos==cn+1)
        {
                head=insert_end(head);
        }
        else
        {
                printf("Enter value for node:");
                scanf("%d",&p->info);
                p->next=NULL;

                t=head;
                for(i=1;i<pos-1;i++)
                        t=t->next;
                p->next=t->next;
                t->next=p;
        }
}
else
```

```
        {
                printf("You Entered a illegal position!\n");
        }
        return head;
}
 node *del(node *head)
 {
        int ch;
        printf("1.Delete first\n");
        printf("2.Delete last\n");
        printf("3.Delete in between\n");
        printf("4.Roll back to menu\n");
        printf("Enter choice:");
        scanf("%d",&ch);

        switch(ch)
        {
                case 1:         head=del_beg(head);
                                        break;
                case 2:         head=del_end(head);
                                        break;
                case 3:         head=del_btw(head);
                                        break;
                case 4:         break;
                default: printf("Wrong choice!\n");
        }
```

```
            return head;
 }

 node *del_beg(node *head)
 {
    node *t;
            if(head==NULL)
            {
                        printf("List is empty!\n");
            }
            else
            {
        t=head;
        head=head->next;

        t->next=NULL;
        free(t);
    }
    return head;

 }
node *del_end(node *head)
{
   node *t,*u;
           if(head==NULL)
           {
                       printf("List is empty!\n");
```

```
            }
            else
            {
        t=head;
        u=t->next;
        while(u->next!=NULL)
        {
           t=t->next;
           u=u->next;
        }
        t->next=NULL;
        free(u);
    }
    return head;
}

void display(node *head)
{
            node *t;
            if(head==NULL)
            {
                        printf("List is empty!\n");
            }
            else
            {
```

```
            t=head;
            while(t!=NULL)
            {
                  printf("%d ",t->info);
                  t=t->next;
            }
            printf("\n");
   }
}
node *del_btw(node *head)
{
   node *t,*u;
   int pos,cn,i;
         if(head==NULL)
         {
               printf("List is empty!\n");
         }
         else
         {
      printf("Enter position:");
      scanf("%d",&pos);
      cn=count_nodes(head);
      if(pos>0&&pos<cn+1)
            {
                  if(pos==1)
```

```
		{
			head=del_beg(head);
		}
		else if(pos==cn)
		{
			head=del_end(head);
		}

		else
		{
	t=head;
	u=t->next;
			for(i=1;i<pos-1;i++)
			{
				t=t->next;
				u=u->next;
	}
			t->next=u->next;
			free(u);
		}
	 }

	 else
	 {
		printf("You Entered a illegal position!\n");
	 }
}
```

```
    return head;

}
```

OUTCOME:

1) To program to create, insert and delete a node from a singly linked list. (consider all the cases).