

LOYOLA ACADEMY DEGREE & PG COLLEGE

OLD ALWAL, SECUNDERABAD – 500010

[AN AUTONOMOUS DEGREE COLLEGE AFFILIATED TO OSMANIA UNIVERSITY]

**‘A college with potential for excellence’ by UGC. Re-accredited with
GRADE ‘A’ by NAAC**



DEPARTMENT OF COMPUTER DATA SCIENCE & DATA ANALYTICS ENGINEERING Practical Record : Data Structures Through ‘C’

Certificate

This is to **Certify** that this is a **BONAFIDE RECORD** of the work done in **DATA STRUCTURES THROUGH ‘C’** lab during the **1st YEAR / 2nd SEMESTER** in the academic year 2023-2024.

NAME :

CLASS :

HALL TICKET NUMBER :

INTERNAL EXAMINER

PRINCIPAL

EXTERNAL EXAMINER

INDEX

S NO	PROGRAM	PAGE NUMBERS
1	Program to accept sparse matrix and convert it into 3tuple representation	
2	Program to find sum of sparse matrices and convert to 3 tuple representation	
3	Program to convert a sparse matrix into 3 tuple representation after transposing it	
4	Program to implement stack operations using arrays	
5	Program to implement queue operations using arrays	
6	Program to evaluate a postfix expression	
7	Program to convert an infix expression to postfix expression	
8	Program to convert an infix expression to its prefix form	
9	Program to perform linear search	
10	Program to perform binary search	
11	Program to perform bubble sort	
12	Program to implement merge sort	
13	Program to implement quick sort	
14	Program to implement selection sort	
15	Program to implement insertion sort	
16	Program to create a linked list	
17	Program to add a node at the beginning of the linked list	
18	Program to add a node at the end of the linked list	
19	Program to delete a node in the beginning of the linked list	
20	Program to delete a node in the end of the linked list	
21	Program to delete any node in the linked list	
22	Program to split a given linked list in to two	
23	Program to copy a linked list in to another linked list	
24	Program to add a node before a particular node in the linked list	
25	Program to add a node after a particular node in the linked list	
26	Program to concatenate two linked lists	
27	Program to search an element in the linked list	
28	Program to find the number of elements in the given linked list	
29	Program to implement stack using linked list	
30	Program to implement queue using linked list	

/*PROGRAM TO ACCEPT SPARSE MATRIX AND CONVERT IT IN TO 3 TUPLE REPRESENTATION*/

```
#include<stdio.h>
void main()
{
int a[3][3],i,j,c=0,t[3][3],k=0;
printf("\n enter sparse matrix");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
if(a[i][j]!=0)
c++;
}
}
for(i=0; i<3;i++)
{
for(j=0; j<3;j++)
{
if(a[i][j]!=0)
{
t[k][0]=i;
t[k][1]=j;
t[k][2]=a[i][j];
k++;
}
}
}
printf("\n the 3 tuple representation of the given sparse matrix is:\n");
for(i=0;i<c;i++)
{
for(j=0;j<3;j++)
{
printf("\t%d",t[i][j]);
}
printf("\n");
}
}
OUTPUT:
```

```
enter sparse matrix
1 0 0
0 0 0
0 0 2
```

the 3 tuple representation of the given sparse matrix is:

0	0	1
2	2	2



```
/*PROGRAM TO FIND SUM OF A SPARSE MATRICES AND CONVERT TO 3 TUPPLE  
REPRESENTATION*/
```

```
#include<stdio.h>
void main()
{
int a[3][3],t[3][3],i,j,k=0,ct=0;
printf("\n enter the sparse matrix:");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
if(a[i][j]!=0)
ct++;
}
}
printf("\n the transpose of given sparse matrix is \n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
printf("\t%d",a[j][i]);
}
}
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
if(a[j][i]!=0)
{
t[k][0]=j;
t[k][1]=i;
t[k][2]=a[j][i];
k++;
}
}
}
printf("\n the sparse matrix in 3 tuple representation of the transposed matrix is");
for(i=0;i<ct;i++)
{
for(j=0;j<3;j++)
{
printf("\t%d",t[i][j]);
}
printf("\n");
}
}
```



OUTPUT:

Enter the elements of the 1st sparse matrix

0 0 4

0 0 0

2 0 0

Enter the elements of the 2nd sparse matrix

0 0 0

3 0 0

0 0 2

The resultant sparse matrix is

0 0 4

3 0 0

2 0 2

The non-zero elements in the resultant sparse matrix is 4

The 3 tuple representation of the given sparse matrix is:0

2 4

1 0 3

2 0 2

2 2 2



```
/*PROGRAM TO CONVERT A SPARSE MATRIX IN TO 3-TUPPLE REPRESENTATION  
AFTER TRANSPOSING IT*/
```

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main()
```

```
{
```

```
int a[3][3],t[3][3],i,j,k=0,ct=0;
```

```
clrscr();
```

```
printf("\n enter the sparse matrix elements:\n");
```

```
for(i=0;i<3;i++)
```

```
{
```

```
for(j=0;j<3;j++)
```

```
{
```

```
scanf("%d",&a[i][j]);
```

```
if(a[i][j]!=0)
```

```
ct++;
```

```
}
```

```
}
```

```
printf("\n the transpose of given matrix is");
```

```
for(i=0;i<3;i++)
```

```
{
```

```
for(j=0;j<3;j++)
```

```
{
```

```
printf("\t%d",a[j][i]);
```

```
}
```

```
printf("\n");
```

```
}
```

```
for(i=0;i<3;i++)
```

```
{
```

```
for(j=0;j<3;j++)
```

```
{
```

```
if(a[j][i]!=0)
```

```
{
```

```
t[k][0]=j;
```

```
t[k][1]=i;
```

```
t[k][2]=a[j][i];
```

```
k++;
```

```
}
```

```
}
```

```
}
```

```
printf("\n the 3-tuple representation of the matrix is");
```

```
for(i=0;i<ct;i++)
```

```
{
```

```
for(j=0;j<3;j++)
```

```
{
```

```
printf("\t%d",t[i][j]);
```

```
}
```

```
printf("\n");
```



```
}  
}
```

OUTPUT:

Enter the elements of the sparse matrix:

0 0 7

0 0 0

5 0 0

The transpose of given sparse matrix is

0 0 5

0 0 0

7 0 0

The 3 tuple representation of the transposed sparse matrix is

0 5

0 2 7



```
/*PROGRAM TO IMPLEMENT STACK OPERATIONS USING ARRAYS*/
```

```
#include<stdio.h>
void push(int);
void pop();
void display();
void option();
int a[20],top=-1;
void main()
{
int n,ch,item;
printf("enter the size of the stack");
scanf("%d",&n);
do
{
option();
printf("enter your choice");
scanf("%d",&ch);
switch(ch)
{
case 1:
if(top>=n-1)
printf("stack is full \n");
else
{
printf("enter the item");
scanf("%d",&item);
push(item);
}
break;
case 2:
if(top<0)
printf("stack is empty \n");
else
pop();
break;
case 3:
if(top<0)
printf("stack is empty \n");
else
display();
break;
case 4:
exit();
break;
}
}
while((ch>0)&&(ch<5));
```




```

}
void push(int item)
{
top++;
a[top]=item;
}
void pop()
{
printf("the popped element is %d \n",a[top]);
top--;
}
void display()
{
int i;
for(i=0; i<=top;i++)
printf("%d \n",a[i]);
}
void option()
{
printf("1.push \n");
printf("2.pop \n");
printf("3.display \n");
printf("4.exit \n");
}

```

OUTPUT:

Enter the size of the stack 4

1. Push
2. Pop
3. Display
4. Exit

Enter your choice 1

Enter the item 10

1. Push
2. Pop
3. Display
4. Exit

Enter your choice 1

Enter the item 20

1. Push
2. Pop
3. Display
4. Exit

Enter your choice

Enter the item 30

1. Push
2. Pop
3. Display



4. Exit

Enter your choice 1

Enter your item 40

1. Push

2. Pop

3. Display

4. Exit

Enter your choice 2

The popped element is 40

1. Push

2. Pop

3. Display

4. Exit

Enter your choice 2

The popped element is 30

1. Push

2. Pop

3. Display

4. Exit

Enter your choice 2

The popped element is 20

1. Push

2. Pop

3. Display

4. Exit

Enter your choice 2

The popped element is 10

1. Push

2. Pop

3. Display

4. Exit

Enter your choice 3

Stack is empty

1. Push

2. Pop

3. Display

4. Exit

Enter your choice 4



```
/*PROGRAM TO IMPLEMENT QUEUE OPERATIONS USING ARRAYS*/
```

```
#include<stdio.h>
void insert(int);
void del();
void display();
void option();
int queue[20],item,f=0,r=0;
void main()
{
    int n,ch;
    printf("enter the size of the queue");
    scanf("%d",&n);
    do
    {
        option();
        printf("enter your choice");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                if(r>=n)
                    printf("queue is full \n");
                else
                {
                    printf("enter the item");
                    scanf("%d",&item);
                    insert(item);
                }
                break;
            case 2:
                if(r==f)
                    printf("queue is empty \n");
                else
                    del();
                break;
            case 3:
                if(r==f)
                    printf("stack is empty \n");
                else
                    display();
                break;
            case 4:
                exit();
                break;
        }
    }
    while((ch>0)&&(ch<5));
```



```

}
void insert(int item)
{
queue[r]=item;
r++;
}
void del()
{
printf("the deleted element is %d \n",queue[f]);
f++;
}
void display()
{
int i;
for(i=f;i<r;i++)
printf("%d \n",queue[i]);
}
void option()
{
printf("1.insert \n");
printf("2.del \n");
printf("3.display \n");
printf("4.exit \n");
}

```

OUTPUT:

Enter the size of the queue 3

1.Insert
2.delete
3.display
4.exit

Enter your choice 1

Enter the item:5

1.insert
2.delete
3.display
4.exit

Enter your choice 1

Enter the item:6

1.insert
2.delete
3.display
4.exit

Enter your choice 1

Enter the item:7

1.insert
2.delete
3.display
4.exit



Enter your choice 2

The deleted element is 5

1.insert

2.delete

3.display

4.exit

Enter your choice 2

The deleted element is 6

1.insert

2.delete

3.display

4.exit

Enter your choice 2

The deleted element is 7

1.insert

2.delete

3.display

4.exit

Enter your choice 3

Queue is empty

1.insert

2.delete

3.display

4.exit

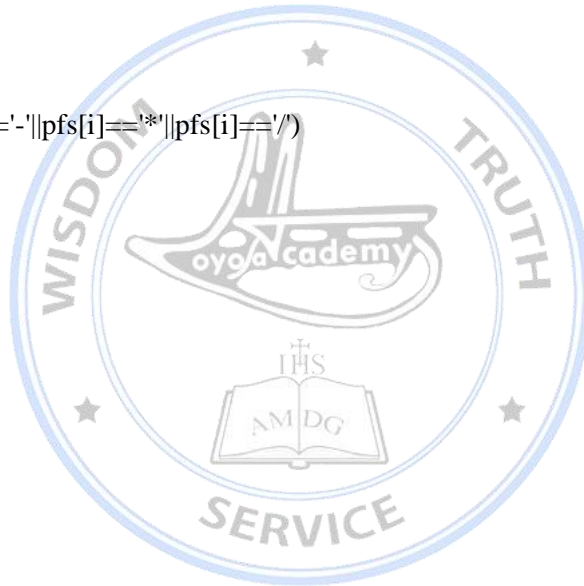
Enter your choice 4



```

/*PROGRAM TO EVALUATE A POSTFIX EXPRESSION*/
#include<stdio.h>
pushpevint poppev();
void pushpev(int);
int top=-1;
char pfs[20];
int pev[20];
void main()
{
int i,n1,v1,v2,n;
printf("\n enter the size:");
scanf("%d",&n);
printf("enter the postfix expression:");
for(i=0;i<n;i++)
scanf("%c",&pfs[i]);
for(i=0;i<n;i++)
{
if(pfs[i]>='0'&&pfs[i]<='9')
{
n1=(int)(pfs[i]-'0');
pushpev(n1);
}
else if(pfs[i]=='+'||pfs[i]=='-'||pfs[i]=='*'||pfs[i]=='/')
{
switch(pfs[i])
{
case '+':v1=poppev();
v2=poppev();
pushpev(v2+v1);
break;
case '-':v1=poppev();
v2=poppev();
pushpev(v2-v1);
break;
case '*':v1=poppev();
v2=poppev();
pushpev(v2*v1);
break;
case '/':v1=poppev();
v2=poppev();
pushpev(v2/v1);
break;
}
}
}
printf("\n the final result is %d",pev[top]);
}
void pushpev(int temp)
{
top++;
pev[top]=temp;
}
int poppev()
{
int temp;

```



```
temp=pev[top];  
top--;  
return(temp);  
}
```

OUTPUT:

Enter the size:6

Enter the postfix expression:23+4-

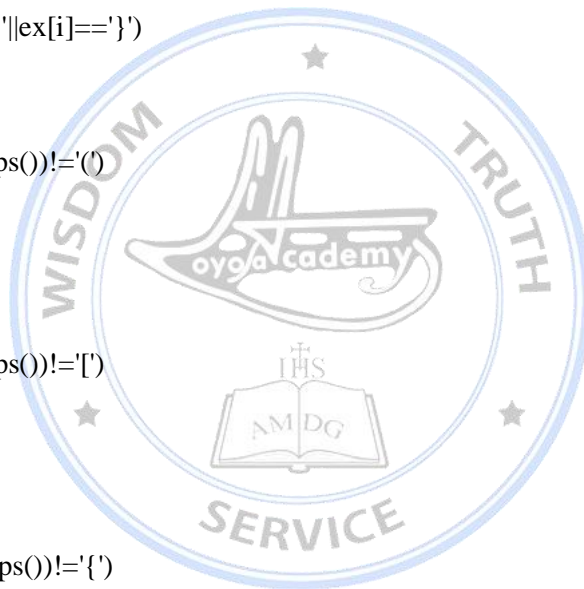
The final result is 1



```

/*PROGRAM TO CONVERT AN INFIX EXPRESSION TO POSTFIX EXPRESSION*/
#include<stdio.h>
void pushops(char);
void pushpfs(char);
char popops();
char ex[20],ops[20],pfs[20],sym;
int i,t1=-1,t2=-1,n;
void main()
{
printf("\n enter the no. of characters");
scanf("%d",&n);
for(i=0;i<n;i++)
scanf("%c",&ex[i]);
for(i=0;i<n;i++)
{
if(ex[i]=='('||ex[i]=='{'||ex[i]=='['||ex[i]=='+'||ex[i]=='-'||ex[i]=='*'||ex[i]=='/')
pushops(ex[i]);
else if(ex[i]>='a'&&ex[i]<='z'||ex[i]>='A'&&ex[i]<='Z')
pushpfs(ex[i]);
else if(ex[i]==')'||ex[i]=='}'||ex[i]==']')
{
switch(ex[i])
{
case ')':while((sym=popops())!='(')
{
t2++;
pfs[t2]=sym;
}
break;
case ']':while((sym=popops())!='[')
{
t2++;
pfs[t2]=sym;
}
break;
case '}':while((sym=popops())!='{')
{
t2++;
pfs[t2]=sym;
}
break;
}
}
}
printf("\n the post fix exp:\n");
for(i=0;i<n;i++)
printf("%c",pfs[i]);
}
void pushops(char temp)
{
t1=t1+1;
ops[t1]=temp;
}
void pushpfs(char temp)
{

```




```
t2=t2+1;
pfs[t2]=temp;
}
char popops()
{
char temp;
temp=ops[t1];
t1--;
return(temp);
}
```

OUTPUT:

Enter the number of characters 14

[(a+b)*(c+d)]

The postfix exp:ab+cd+*



```

/*PROGRAM TO CONVERT AN INFIX EXPRESSION TO ITS PREFIX FORM*/
#include<stdio.h>
void pushops(char);
void pushpfs(char);
char popops();
char ex[20],rex[20],ops[20],pfs[20],sym;
int i,t1=-1,j,n;
void main()
{
printf("\n enter the no. of characters");
scanf("%d",&n);
printf("\n enter the exp:\n");
for(i=0;i<n;i++)
scanf("%c",&ex[i]);
for(i=n-1,j=0;i>=0;i--,j++)
rex[j]=ex[i];
for(i=0;i<n;i++)
{
if(rex[i]=='('||rex[i]=='{'||rex[i]=='['||rex[i]=='+'||rex[i]=='-'||rex[i]=='*'||rex[i]=='/')
pushops(rex[i]);
else if(rex[i]>='a'&&rex[i]<='z'||rex[i]>='A'&&rex[i]<='Z')
pushpfs(rex[i]);
else if(rex[i]=='('||rex[i]=='{'||rex[i]=='[')
switch(rex[i])
{
case '(':while((sym=popops())!='))
{
t2++;
pfs[t2]=sym;
}
break;
case '[':while((sym=popops())!='])
{
t2++;
pfs[t2]=sym;
}
break;
case '{':while((sym=popops())!='}')
{
t2++;
pfs[t2]=sym;
}
break;
}
}
printf("\n the prefix exp:\n");
for(i=n-1;i>=0;i--)
printf("%c",pfs[i]);
}
void pushops(char temp)
{
t1=t1+1;
ops[t1]=temp;
}
void pushpfs(char temp)

```

```
{  
t2=t2+1;  
pfs[t2]=temp;  
}  
char popops()  
{  
char temp;  
temp=ops[t1];  
t1--;  
return(temp);  
}
```

OUTPUT:

Enter the number of characters 6

Enter the exp:(a+b)

The prefix exp:

+ab



```
/*PROGRAM TO PERFORM LINEAR SEARCH*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
int a[10],n,i,c=0;
```

```
printf("enter the elements of the array");
```

```
for(i=0;i<10;i++)
```

```
scanf("%d",&a[i]);
```

```
printf("enter the element to search");
```

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

```
for(i=0;i<10;i++)
```

```
{
```

```
if(a[i]==n)
```

```
{
```

```
c++;
```

```
break;
```

```
}
```

```
}
```

```
if(c>0)
```

```
printf("%d is found at position %d",n,i+1);
```

```
else
```

```
printf("element not found");
```

```
}
```

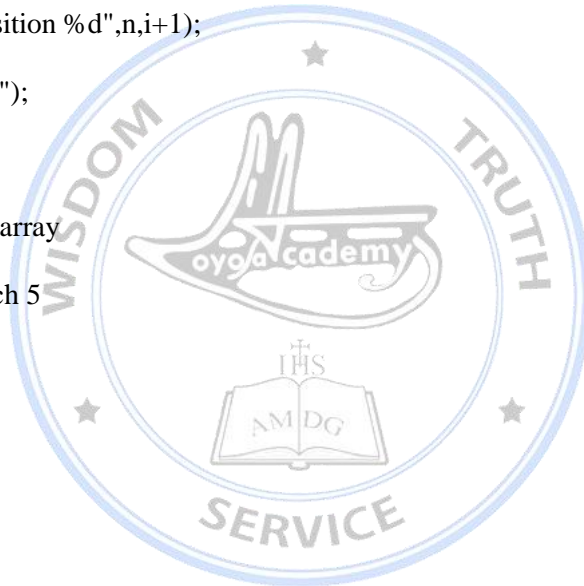
```
OUTPUT:
```

```
Enter the elements of the array
```

```
0 1 2 3 4 5 6 7 8 9
```

```
Enter the element to search 5
```

```
5 is found at position 6
```



```
/*PROGRAM TO PERFORM BINARY SEARCH*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
int a[10],x,i,n,flag,low,mid,high;
```

```
printf("enter the size of the array:");
```

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

```
printf("enter the elements of the array:");
```

```
for(i=0;i<10;i++)
```

```
scanf("%d",&a[i]);
```

```
printf("enter the element to search:");
```

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

```
low=0;
```

```
high=n-1;
```

```
while(low<=high)
```

```
{
```

```
mid=(low+high)/2;
```

```
if(x<a[mid])
```

```
high=mid-1;
```

```
else if(x>a[mid])
```

```
low=mid+1;
```

```
else if(x==a[mid])
```

```
{
```

```
flag=0;
```

```
break;
```

```
}
```

```
}
```

```
if(flag!=0)
```

```
printf("search is unsuccessful");
```

```
else
```

```
printf("search is successful");
```

```
}
```

```
OUTPUT:
```

Enter the size of the array:10

Enter the elements of the array:

0 1 2 3 4 5 6 7 8 9

Enter the element to search:6

Search is successful



```
/*PROGRAM TO PERFORM BUBBLE SORT*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
int a[10],i,j,temp,n;
```

```
printf("enter the size of the array");
```

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

```
printf("enter the elements in to the array:");
```

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

```
scanf("%d",&a[i]);
```

```
printf("the sorted array is:\n");
```

```
for(i=0;i<n-1;i++)
```

```
{
```

```
for(j=0;j<n-1;j++)
```

```
{
```

```
if(a[j]>a[j+1])
```

```
{
```

```
temp=a[j];
```

```
a[j]=a[j+1];
```

```
a[j+1]=temp;
```

```
}
```

```
}
```

```
}
```

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

```
printf("%d\n",a[i]);
```

```
}
```

```
OUTPUT:
```

Enter the size of the array 5

Enter the elements in to the array: 11 9 13 10 8

The sorted array is

8

9

10

11

13



```

/*PROGRAM TO IMPLEMENT MERGESORT*/
#include<stdio.h>
void mergesort(int a[],int b[],int c[],int n1,int n2,int n3);
int a[10],b[10],c[10],n,n1,n2,n3,i;
void main()
{
printf("enter the size of the array A:");
scanf("%d",&n1);
printf("enter the sorted elements in A \n");
for(i=0;i<n1;i++)
scanf("%d",&a[i]);
printf("enter the size of the array B:");
scanf("%d",&n2);
printf("enter the sorted elements in B \n");
for(i=0;i<n2;i++)
scanf("%d",&b[i]);
n3=(n1+n2);
mergesort(a,b,c,n1,n2,n3);
printf("elements after merging:\n");
for(i=0;i<n3;i++)
printf("%d\n",c[i]);
}
void merge(int a[],int b[],int c[],int n1,int n2,int n3)
{
int apoint,bpoint,cpoint;
int alimit,blimit,climit;
alimit=n1-1;
blimit=n2-1;
apoint=0;
bpoint=0;
for(cpoint=0;apoint<=alimit&&bpoint<=blimit;cpoint++)
{
if(a[apoint]<b[bpoint])
c[cpoint]=a[apoint++];
else
c[cpoint]=b[bpoint++];
}
while(apoint<=alimit)
c[cpoint++]=a[apoint++];
while(bpoint<=blimit)
c[cpoint++]=b[bpoint++];
}

```

OUTPUT:

Enter the size of the array A:5
Enter the sorted elements in A
1 3 5 7 9
Enter the size of the array B:5
Enter the sorted elements in B
2 4 6 8 10
Elements after merging:
1
2
3
4

5
6
7
8
9
10




```
/*PROGRAM TO IMPLEMENT QUICKSORT*/
```

```
#include<stdio.h>
```

```
void quick(int x[],int lb,int ub);
```

```
void main()
```

```
{
```

```
int x[20],i,n,a;
```

```
printf("enter the size of the array");
```

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

```
printf("enter the elements of the array \n");
```

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

```
scanf("%d",&x[i]);
```

```
quick(x,0,n-1);
```

```
printf("array after sorting \n");
```

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

```
printf("%d\n",x[i]);
```

```
}
```

```
void quick(int x[],int lb,int ub)
```

```
{
```

```
int a,n,up,down,temp;
```

```
a=x[lb];
```

```
up=ub;
```

```
down=lb;
```

```
while(down<up)
```

```
{  
while(x[down]<=a&&down<ub)
```

```
down++;
```

```
while(x[up]>a)
```

```
up--;
```

```
if(down<up)
```

```
{
```

```
temp=x[down];
```

```
x[down]=x[up];
```

```
x[up]=temp;
```

```
}
```

```
}
```

```
x[lb]=x[up];
```

```
x[up]=a;
```

```
if(lb<up)
```

```
quick(x,lb,up);
```

```
if(down<ub)
```

```
quick(x,down,ub);
```

```
}
```

```
OUTPUT:
```

Enter the size of the array 5

Enter the elements of the array

5 3 4 1 6

Array after sorting

1

3

4

5

6



```
/*PROGRAM TO IMPLEMENT SELECTION SORT*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
int arr[5];
```

```
int i,j,temp;
```

```
printf("enter the elements to be sorted");
```

```
for(i=0;i<5;i++)
```

```
scanf("%d",&arr[i]);
```

```
printf("selection sort \n");
```

```
printf("\n array before sorting:\n");
```

```
for(i=0;i<=4;i++)
```

```
{
```

```
printf("%d\t",arr[i]);
```

```
for(j=i+1;j<=4;j++)
```

```
{
```

```
if(arr[i]>arr[j])
```

```
{
```

```
temp=arr[i];
```

```
arr[i]=arr[j];
```

```
arr[j]=temp;
```

```
}
```

```
}
```

```
}
```

```
printf("\n \n array after sorting:\n");
```

```
for(i=0;i<=4;i++)
```

```
printf("%d\t",arr[i]);
```

```
}
```

OUTPUT:

Enter the elements to be sorted

20 7 34 86 30

Array before sorting

20 20 34 86 86

Array after sorting

7 20 30 34 86



```
/*PROGRAM TO IMPLEMENT INSERTION SORT*/
```

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main()
```

```
{
```

```
int a[10];
```

```
int i,j,k,n,temp;
```

```
clrscr();
```

```
printf("enter the size of the list");
```

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

```
printf("enter the elements to be sorted");
```

```
{
```

```
for(i=0;i<10;i++)
```

```
{
```

```
scanf("%d",&a[i]);
```

```
}
```

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

```
{
```

```
printf("%d\t",a[i]);
```

```
}
```

```
for(i=1;i<n;i++)
```

```
{
```

```
for(j=0;j<i;j++)
```

```
{
```

```
if(a[j]>a[i])
```

```
{
```

```
temp=a[j];
```

```
a[j]=a[i];
```

```
for(k=i;k>j;k--)
```

```
a[k]=a[k-1];
```

```
a[k+1]=temp;
```

```
}
```

```
}
```

```
}
```

```
printf(" array after sorting \n");
```

```
for(i=0;i<10;i++)
```

```
{
```

```
printf("%d \t",a[i]);
```

```
}
```

```
getch();
```

```
}
```

OUTPUT:

Enter the elements to be sorted

5 7 9 2 6

Array before sorting

5 7 9 2 6

Array after sorting

2 5 6 7 9



```

/*PROGRAM TO CREATE LINKED LIST*/
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp;
    char ch='y';
    int b;
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);
}

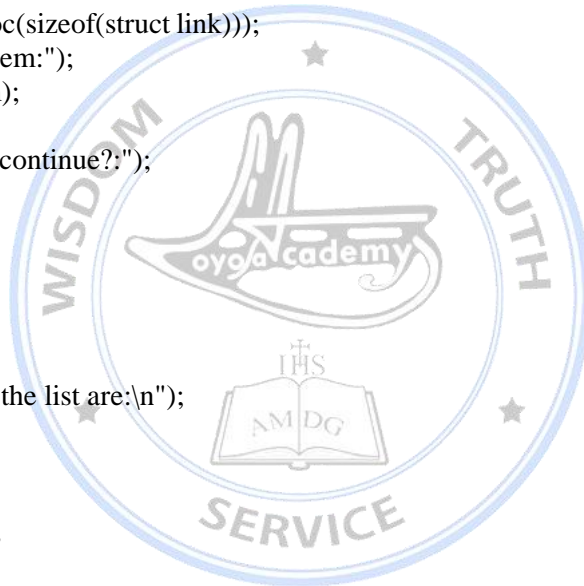
```

OUTPUT:

```

Enter the first element of the list:1
Enter the next item:2
Do you want to continue?:y
Enter the next item:3
Do you want to continue?:n
Elements in the list are:
1->2->3->4->NULL

```



```

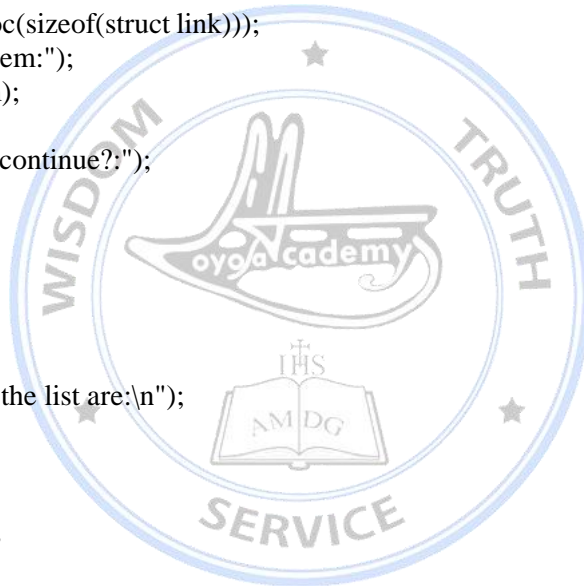
/*PROGRAM TO ADD A NODE AT THE BEGINNING OF THE LINKED LIST*/
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp;
    char ch='y';
    int b;
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);

    printf("\n enter the element to be added in the beginning");
    temp=((struct link*)malloc(sizeof(struct link)));
    scanf("%d",&temp->item);
    temp->ptr=head;
    head=temp;
    cur=head;
    printf("\n the elements of the list after adding the node in the beginning");
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);
}

```

OUTPUT:

Enter the first element of the list:1



Enter the next item:2
Do you want to continue?:y
Enter the next item:3
Do you want to continue?:n
Elements in the list are:1->2->3->NULL
Enter the element to be added in the beginning 5
The elements of the list after adding the node in the beginning
5->1->2->3->NULL



```
/*PROGRAM TO ADD A NODE AT THE END OF THE LIST*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
struct link
```

```
{
```

```
int item;
```

```
struct link*ptr;
```

```
};
```

```
struct link*head,*cur,*temp;
```

```
char ch='y';
```

```
int b;
```

```
head=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the first element of the list:");
```

```
scanf("%d",&head->item);
```

```
head->ptr=NULL;
```

```
cur=head;
```

```
do
```

```
{
```

```
temp=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the next item:");
```

```
scanf("%d",&temp->item);
```

```
temp->ptr=NULL;
```

```
printf("\n do you want to continue?:");
```

```
ch=getche();
```

```
cur->ptr=temp;
```

```
cur=temp;
```

```
}
```

```
while(ch=='y');
```

```
cur->ptr=NULL;
```

```
printf("\n the elements in the list are:\n");
```

```
cur=head;
```

```
while(cur->ptr!=NULL)
```

```
{
```

```
printf("%d->",cur->item);
```

```
cur=cur->ptr;
```

```
}
```

```
printf("%d->NULL",cur->item);
```

```
printf("\n enter the element to be added at the end of the list");
```

```
temp=((struct link*)malloc(sizeof(struct link)));
```

```
scanf("%d",&temp->item);
```

```
temp->ptr=NULL;
```

```
cur->ptr=temp;
```

```
cur=head;
```

```
printf("\n the element after adding a node at the end");
```

```
while(cur->ptr!=NULL)
```

```
{
```

```
printf("%d->",cur->item);
```

```
cur=cur->ptr;
```

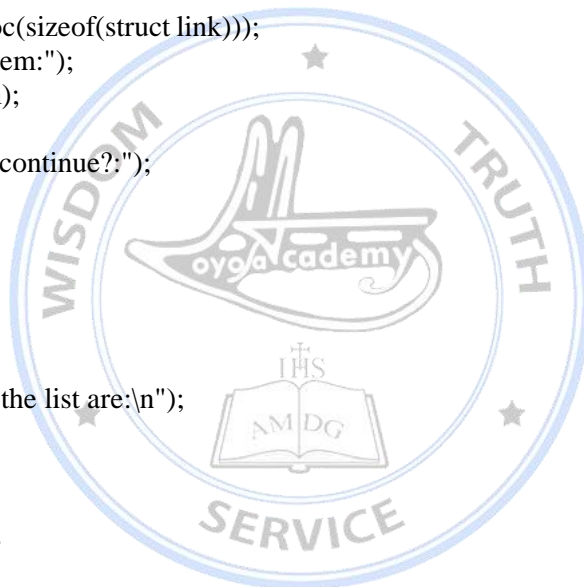
```
}
```

```
printf("%d->NULL",cur->item);
```

```
}
```

OUTPUT:

Enter the first element in the list:1



Enter the next item:2

Do you want to continue?;y

Enter the next item:3

Do you want to continue?:n

The elements in the list are:1->2->->3->NULL

Enter the element to be added at the end of the list 5

The elements after adding a node at the end:

1->2->3->5->NULL




```
/*PROGRAM TO DELETE A NODE IN THE BEGINNING*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
struct link
```

```
{
```

```
int item;
```

```
struct link*ptr;
```

```
};
```

```
struct link*head,*cur,*temp;
```

```
char ch='y';
```

```
int b;
```

```
head=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the first element of the list:");
```

```
scanf("%d",&head->item);
```

```
head->ptr=NULL;
```

```
cur=head;
```

```
do
```

```
{
```

```
temp=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the next item:");
```

```
scanf("%d",&temp->item);
```

```
temp->ptr=NULL;
```

```
printf("\n do you want to continue?:");
```

```
ch=getche();
```

```
cur->ptr=temp;
```

```
cur=temp;
```

```
}
```

```
while(ch=='y');
```

```
cur->ptr=NULL;
```

```
printf("\n the elements in the list are:\n");
```

```
cur=head;
```

```
while(cur->ptr!=NULL)
```

```
{
```

```
printf("%d->",cur->item);
```

```
cur=cur->ptr;
```

```
}
```

```
printf("%d->NULL",cur->item);
```

```
cur=head;
```

```
head=cur->ptr;
```

```
cur->ptr=NULL;
```

```
cur=head;
```

```
printf("\n the modified list is:");
```

```
while(cur->ptr!=NULL)
```

```
{
```

```
printf("%d->",cur->item);
```

```
cur=cur->ptr;
```

```
}
```

```
printf("%d->NULL",cur->item);
```

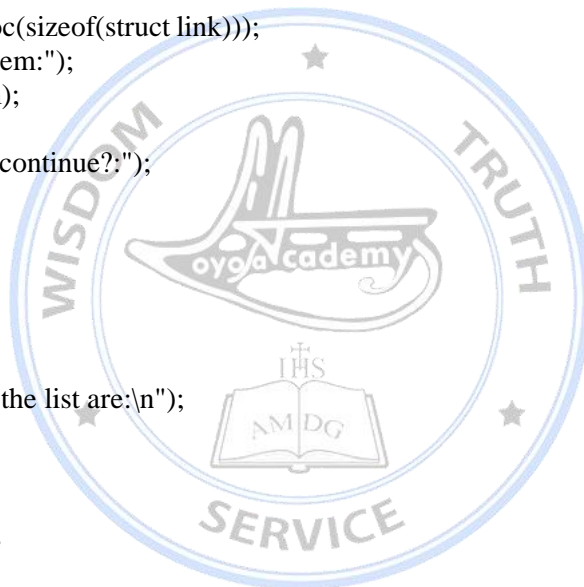
```
}
```

```
OUTPUT:
```

```
Enter the first element in the list:1
```

```
Enter the next item:2
```

```
Do you want to continue?:y
```



Enter the next item:3

Elements in the list are:1->2->3->NULL

The modified list is:

2->3->NULL



```
/*PROGRAM TO DELETE A NODE IN THE END*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
struct link
```

```
{
```

```
int item;
```

```
struct link*ptr;
```

```
};
```

```
struct link*head,*cur,*temp;
```

```
char ch='y';
```

```
int b;
```

```
head=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the first element of the list:");
```

```
scanf("%d",&head->item);
```

```
head->ptr=NULL;
```

```
cur=head;
```

```
do
```

```
{
```

```
temp=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the next item:");
```

```
scanf("%d",&temp->item);
```

```
temp->ptr=NULL;
```

```
printf("\n do you want to continue?:");
```

```
ch=getche();
```

```
cur->ptr=temp;
```

```
cur=temp;
```

```
}
```

```
while(ch=='y');
```

```
cur->ptr=NULL;
```

```
printf("\n the elements in the list are:\n");
```

```
cur=head;
```

```
while(cur->ptr!=NULL)
```

```
{
```

```
printf("%d->",cur->item);
```

```
cur=cur->ptr;
```

```
}
```

```
printf("%d->NULL",cur->item);
```

```
cur=head;
```

```
while(cur->ptr->ptr!=NULL)
```

```
{
```

```
cur=cur->ptr;
```

```
}cur->ptr=NULL;
```

```
cur=head;
```

```
printf("\n the modified list is:");
```

```
while(cur->ptr!=NULL)
```

```
{
```

```
printf("%d->",cur->item);
```

```
cur=cur->ptr;
```

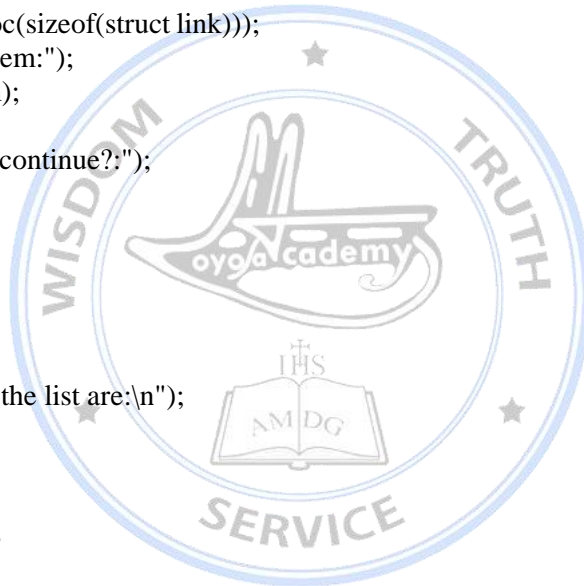
```
}
```

```
printf("%d->NULL",cur->item);
```

```
}
```

```
OUTPUT:
```

```
Enter the first element in the list:1
```



Enter the next item:2

Do you want to continue?:y

Enter the next item:3

Do you want to continue?:n

The elements in the list are:1->2->3->NULL

The modified list is:

1->2->NULL

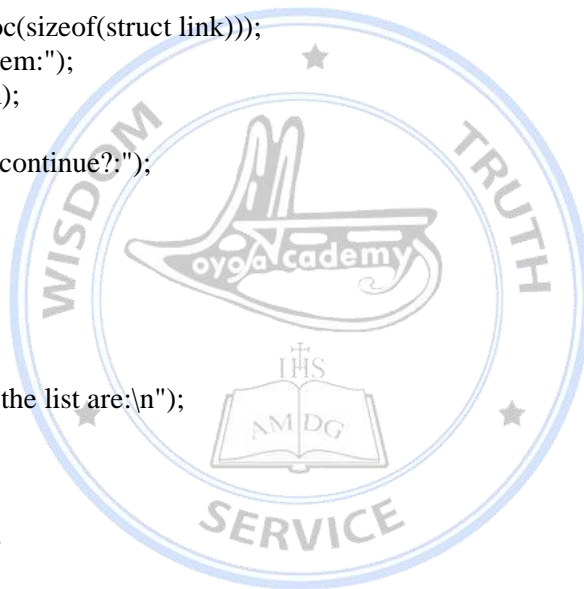


```

/*PROGRAM TO DELETE ANY NODE IN THE LIST*/
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp;
    char ch='y';
    int b;
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);

    printf("\n enter the value of the node to be deleted");
    scanf("%d",&b);
    cur=head;
    while(cur->ptr!=NULL)
    {
        if(cur->ptr->item==b)
        {
            cur->ptr=cur->ptr->ptr;
            break;
        }
        cur=cur->ptr;
    }
    cur=head;
    printf("the modified list is");
    while(cur->ptr!=NULL)
    {

```



```
printf("%d->",cur->item);  
cur=cur->ptr;  
}  
printf("%d->NULL",cur->item);  
}
```

OUTPUT:

Enter the first element in the list:1

Enter the next item:2

Do you want to continue?:y

Enter the next item:3

Do you want to continue?:n

The elements in the list are:1->2->3->NULL

Enter the value of the node to be deleted 2

The modified list is 1->3->NULL

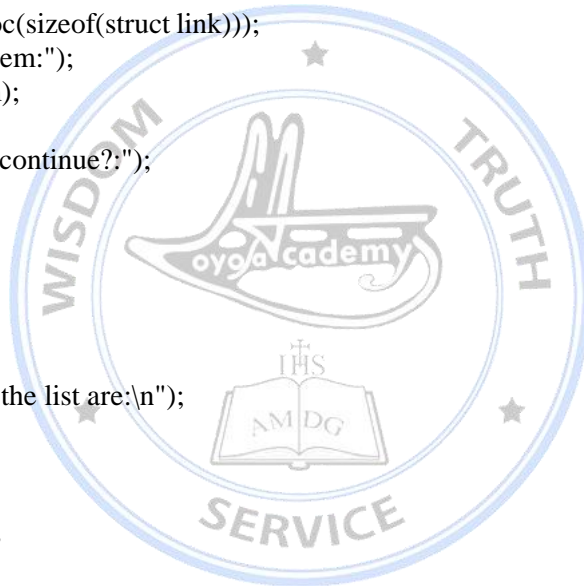


```

/*PROGRAM TO SPLIT A GIVEN LINKED LIST IN TO TWO*/
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp, *head1,*cur1;
    char ch='y';
    int b;
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);

    cur=head;
    printf("enter the value of the node at which you want to split");
    scanf("%d",&b);
    while(cur->ptr!=NULL)
    {
        if(cur->ptr->item==b)
        {
            head1=cur->ptr;
            cur->ptr=NULL;
        }
        cur=cur->ptr;
    }
    printf("the first list is \n");
    cur=head;
    while(cur->ptr!=NULL)
    {

```



```

printf("%d->",cur->item);
cur=cur->ptr;
}
printf("%d->NULL",cur->item);
printf("\n the 2nd list is \n");
cur1=head1;
while(cur1->ptr!=NULL)
{
printf("%d->",cur1->item);
cur1=cur1->ptr;
}
printf("%d->NULL",cur1->item);
}

```

OUTPUT:

Enter the first element in the list:1

Enter the next item:2

Do you want to continue?:y

Enter the next item:3

Do you want to continue?:n

The elements in the list are:1->2->->3->NULL

Enter the value of the node at which you want to split 2

The first list is 1->NULL

The second list is 2->3->NULL

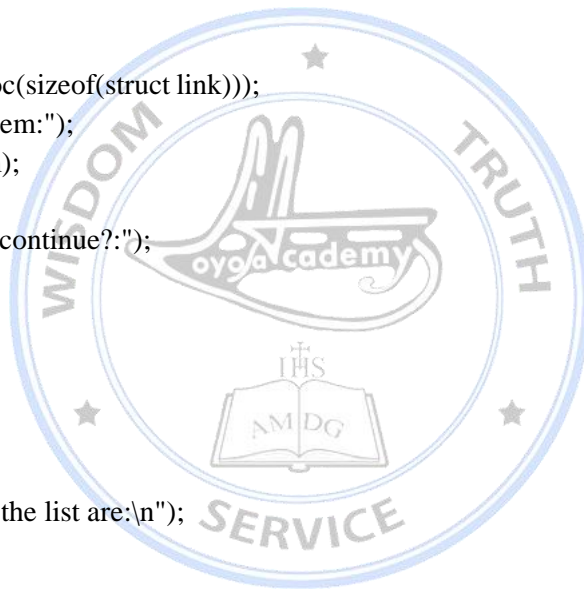



```

/*PROGRAM TO COPY A LINKED LIST IN TO ANOTHER LIST*/
#include<stdio.h>
void main()
{
struct link
{
int item;
struct link*ptr;
};
struct link*head,*cur,*temp,*head1,*cur1;
char ch='y';
head=((struct link*)malloc(sizeof(struct link)));
printf("\n enter the first element of the list:");
scanf("%d",&head->item);
head->ptr=NULL;
cur=head;
do
{
temp=((struct link*)malloc(sizeof(struct link)));
printf("\n enter the next item:");
scanf("%d",&temp->item);
temp->ptr=NULL;
printf("\n do you want to continue?:");
ch=getche();
cur->ptr=temp;
cur=temp;
}
while(ch=='y');
cur->ptr=NULL;
printf("\n the elements in the list are:\n");
cur=head;
while(cur->ptr!=NULL)
{
printf("%d->",cur->item);
cur=cur->ptr;
}
printf("%d->NULL",cur->item);

cur=head;
head1=((struct link*)malloc(sizeof(struct link)));
head1->item=cur->item;
head1->ptr=NULL;
cur1=head1;
while(cur->ptr!=NULL)
{
cur=cur->ptr;
temp=((struct link*)malloc(sizeof(struct link)));
temp->item=cur->item;

```



```
temp->ptr=NULL;
cur1->ptr=temp;
cur1=temp;
}
printf("\n elements in the copied list are:\n");
cur1=head1;
while(cur1->ptr!=NULL)
{
printf("%d->",cur1->item);
cur1=cur1->ptr;
}
printf("%d->NULL",cur->item);
}
```

OUTPUT:

Enter the first element of the list:1

Enter the next item:2

Do you want to continue?:y

Enter the next item:3

Do you want to continue?:n

The elements in the list are:1->2->3->NULL

Elements in the copied list are:1->2->3->NULL



```

/*PROGRAM TO ADD A NODE BEFORE A PARTICULAR NODE IN THE LINKED LIST*/
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp;
    char ch='y';
    int b;
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);

    printf("\n node before which the element to be added");
    scanf("%d",&b);
    printf("\n enter the element");
    temp=((struct link*)malloc(sizeof(struct link)));
    scanf("%d",&temp->item);
    cur=head;
    while(cur->ptr!=NULL)
    {
        if(cur->ptr->item==b)
        {
            temp->ptr=cur->ptr;
            cur->ptr=temp;
            break;
        }
        cur=cur->ptr;
    }
}

```

```

cur=head;
printf("\n the modified list is:\n");
while(cur->ptr!=NULL)
{
printf("%d->",cur->item);
cur=cur->ptr;
}
printf("%d->NULL",cur->item);
}

```

OUTPUT:

Enter the first element in the list:1

Enter the next item:2

Do you want to continue?:y

Enter the next item:3

Do you want to continue?:n

The elements in the list are 1->2->3->NULL

Node before which the element to be added 2

Enter the element 7

The modified list is 1->7->2->3->NULL

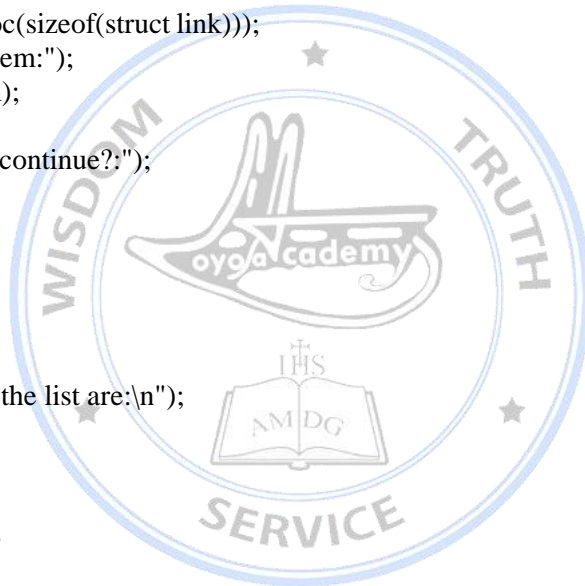


```

/*PROGRAM TO ADD A NODE AFTER A PARTICULAR NODE IN THE LINKED LIST*/
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp;
    char ch='y';
    int b;
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);

    printf("\n node after which the element to be added");
    scanf("%d",&b);
    printf("\n enter the element");
    temp=((struct link*)malloc(sizeof(struct link)));
    scanf("%d",&temp->item);
    cur=head;
    while(cur->ptr!=NULL)
    {
        if(cur->item==b)
        {
            temp->ptr=cur->ptr;
            cur->ptr=temp;
            break;
        }
        cur=cur->ptr;
    }
}

```



```
cur=head;
printf("\n the modified list is:\n");
while(cur->ptr!=NULL)
{
printf("%d->",cur->item);
cur=cur->ptr;
}
printf("%d->NULL",cur->item);
}
```

OUTPUT:

Enter the first element in the list:1

Enter the next item:2

Do you want to continue?:y

Enter the next item:3

Do you want to continue?:n

The elements in the list are:1->2->3->NULL

Node after which the element to be added 2

Enter the element 5

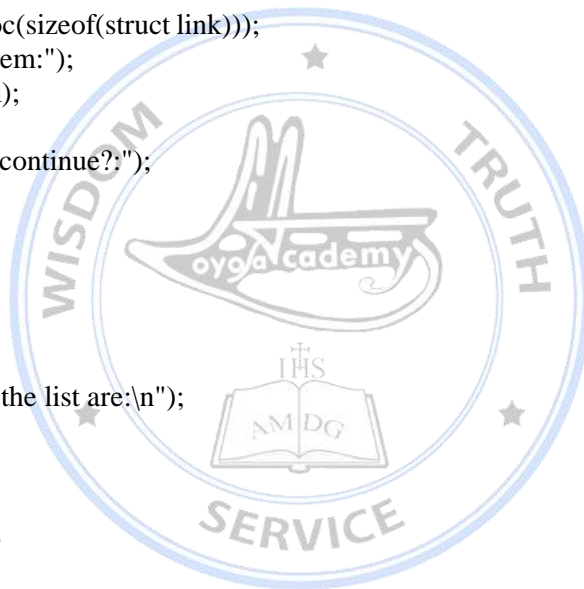
The modified list is:1->2->5->3->NULL



/*PROGRAM TO CONCATENATE TWO LINKED LISTS*/

```
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp,*head1,*cur1;
    char ch='y';
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);

    head1=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head1->item);
    head1->ptr=NULL;
    cur1=head1;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur1->ptr=temp;
        cur1=temp;
    }
}
```



```

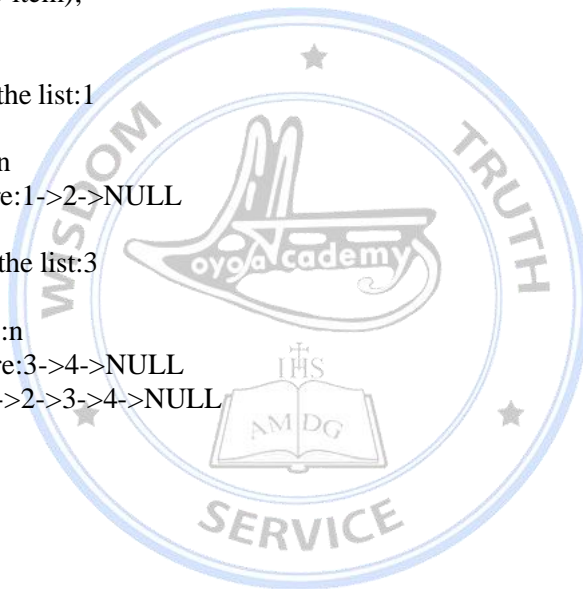
while(ch=='y');
cur1->ptr=NULL;
printf("\n the elements in the list are:\n");
cur1=head1;
while(cur1->ptr!=NULL)
{
printf("%d->",cur1->item);
cur1=cur1->ptr;
}
printf("%d->NULL",cur1->item);
cur->ptr=head1;
printf("\n the concatenated list is:");
cur=head;
while(cur->ptr!=NULL)
{
printf("%d->",cur->item);
cur=cur->ptr;
}
printf("%d->NULL",cur->item);
}

```

OUTPUT:

Enter the first element in the list:1
Enter the next item:2
Do you want to continue:n
The elements in the list are:1->2->NULL

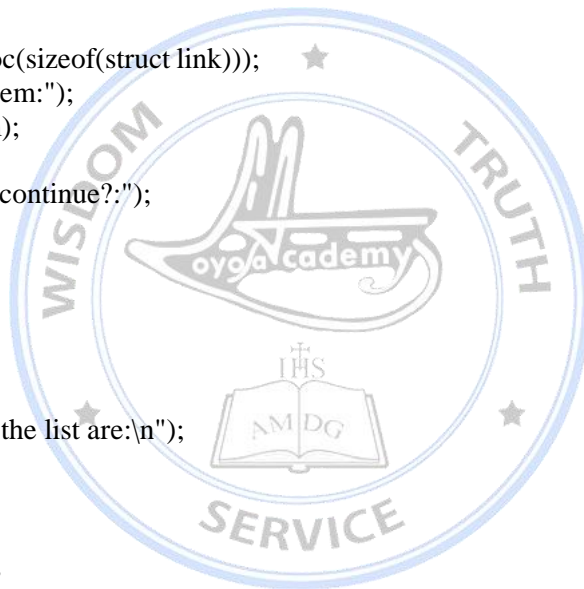
Enter the first element of the list:3
Enter the next item:4
Do you want to continue?:n
The elements in the list are:3->4->NULL
The concatenated list is 1->2->3->4->NULL




```
/*PROGRAM TO SEARCH AN ELEMENT IN THE LIST*/
```

```
#include<stdio.h>
void main()
{
    struct link
    {
        int item;
        struct link*ptr;
    };
    struct link*head,*cur,*temp;
    char ch='y';
    int x,c;
    head=((struct link*)malloc(sizeof(struct link)));
    printf("\n enter the first element of the list:");
    scanf("%d",&head->item);
    head->ptr=NULL;
    cur=head;
    do
    {
        temp=((struct link*)malloc(sizeof(struct link)));
        printf("\n enter the next item:");
        scanf("%d",&temp->item);
        temp->ptr=NULL;
        printf("\n do you want to continue?:");
        ch=getche();
        cur->ptr=temp;
        cur=temp;
    }
    while(ch=='y');
    cur->ptr=NULL;
    printf("\n the elements in the list are:\n");
    cur=head;
    while(cur->ptr!=NULL)
    {
        printf("%d->",cur->item);
        cur=cur->ptr;
    }
    printf("%d->NULL",cur->item);

    printf("\n enter the element to be searched:");
    scanf("%d",&x);
    cur=head;
    while(cur!=NULL)
    {
        if(cur->item==x)
        {
            c=0;
            break;
        }
        cur=cur->ptr;
    }
    if(c==0)
        printf("\n element is found at %u",cur);
    else
```



```
printf("\n element is not found");  
}
```

OUTPUT:

Enter the first element in the list:1

Enter the next item:2

Do you want to continue?:y

Enter the next item:3

Do you want to continue?:y

Enter the next item:4

Do you want to continue?:n

The elements in the list are:1->2->3->4->NULL

Enter the element to be searched:4

Element is found at 2078



```
/*PROGRAM TO FIND THE NUMBER OF ELEMENTS IN THE GIVEN LINKED LIST*/
```

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
struct link
```

```
{
```

```
int item;
```

```
struct link*ptr;
```

```
};
```

```
struct link*head,*cur,*temp;
```

```
char ch='y';
```

```
int ct=0;
```

```
head=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the first element of the list:");
```

```
scanf("%d",&head->item);
```

```
head->ptr=NULL;
```

```
cur=head;
```

```
do
```

```
{
```

```
temp=((struct link*)malloc(sizeof(struct link)));
```

```
printf("\n enter the next item:");
```

```
scanf("%d",&temp->item);
```

```
temp->ptr=NULL;
```

```
printf("\n do you want to continue?:");
```

```
ch=getche();
```

```
cur->ptr=temp;
```

```
cur=temp;
```

```
}
```

```
while(ch=='y');
```

```
cur->ptr=NULL;
```

```
printf("\n the elements in the list are:\n");
```

```
cur=head;
```

```
while(cur->ptr!=NULL)
```

```
{
```

```
printf("%d->",cur->item);
```

```
cur=cur->ptr;
```

```
}
```

```
printf("%d->NULL",cur->item);
```

```
cur=head;
```

```
while(cur!=NULL)
```

```
{
```

```
ct++;
```

```
cur=cur->ptr;
```

```
}
```

```
printf("\n no. of elements is %d",ct);
```

```
}
```

```
OUTPUT:
```

```
Enter the first element in the list:1
```

```
Enter the next item:2
```

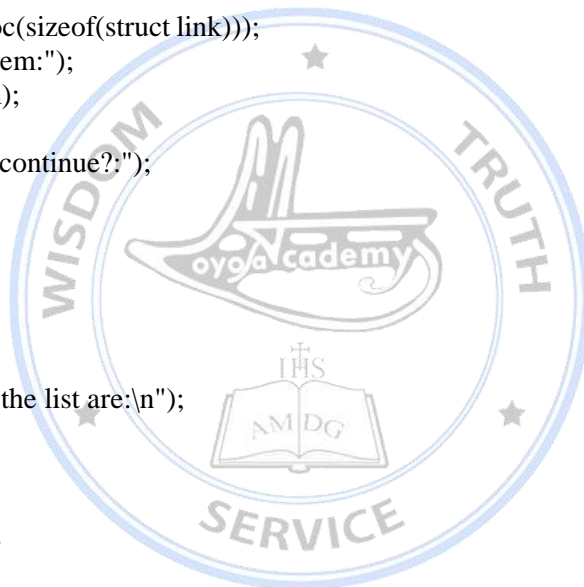
```
Do you want to continue?:y
```

```
Enter the next item:3
```

```
Do you want to continue?:n
```

```
The elements in the list are:1->2->3->NULL
```

```
Number of elements are 3
```



```

/*PROGRAM TO IMPLEMENT STACK USING LINKED LIST*/
#include<stdio.h>
struct link
{
int item;
struct link*ptr;
};
struct link*head,*cur,*temp;
int l,ct=0;
void main()
{
void push(int,int);
void pop();
void display();
int i,item;
printf("enter the size of the stack \n");
scanf("%d",&l);
printf("linked list implementation of a stack \n 1.push \n 2.pop \n 3.display \n 4.peep \n 5.exit");
while(1)
{
printf("\n enter your choice");
scanf("%d",&i);
switch(i)
{
case 1:if(ct==l)
printf("\n stack is full");
else
{
printf("\n enter the item \n");
scanf("%d",&item);
if(ct==0)
push(item,0);
else
push(item,l);
}
break;
case 2:if(ct==0)
{
printf("\n stack is empty");
break;
}
else
pop();
break;
case 3:if(ct==0)
{
printf("\n stack is empty");
break;
}
else
display();
break;
case 4:if(ct==0)
{
printf("\n stack is empty");

```



```

break;
}
else
printf("\n the elements top is pointing to:%d",cur->item);
break;
case 5:exit(1);
}
}
}
void push(int x,int ch)
{
if(ch==0)
{
head=((struct link*)malloc(sizeof(struct link)));
head->item=x;
head->ptr=NULL;
cur=head;
}
else
{
temp=((struct link*)malloc(sizeof(struct link)));
temp->item=x;
temp->ptr=NULL;
cur->ptr=temp;
cur=temp;
}
ct++;
}
void pop()
{
cur=head;
while(cur->ptr!=NULL)
{
cur=cur->ptr;
}
if(ct==1)
printf("\n elements popped from the stack is %d",cur->item);
else
printf("\n elements popped from the stack is %d",cur->ptr->item);
cur->ptr=NULL;
ct--;
}
void display()
{
printf("\n the elements in the list are:");
cur=head;
while(cur->ptr!=NULL)
{
printf("%d->",cur->item);
cur=cur->ptr;
}
printf("%d->NULL",cur->item);
}
}
OUTPUT:

```



Enter the size of the stack:3

The linked list implementation of the stack

1.push

2.pop

3.display

4.exit

Enter your choice:1

Enter the element 10

Enter your choice:1

Enter the element 20

Enter your choice:1

Enter the element 30

Enter your choice:3

The elements in the stack are 10->20->30->NULL

Enter your choice:2

The element popped is 10

Enter your choice:2

The element popped is 20

Enter your choice 2

The element popped is 30

Enter your choice 2

Queue is empty

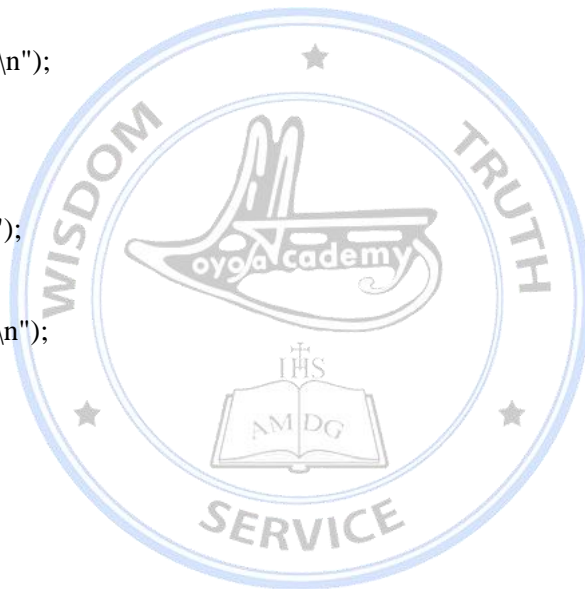
Enter your choice 4



```

/*PROGRAM TO IMPLEMENT QUEUE USING LINKED LIST*/
#include<stdio.h>
struct link
{
int item;
struct link*ptr;
};
struct link *head,*cur,*temp;
int l,ct=0;
void main()
{
void insert(int,int);
void del();
void display();
int i,item;
printf("enter the size of the queue \n");
scanf("%d",&l);
printf("linked list implementation of a queue \n 1.Insertion \n 2.Deletion \n 3.Display \n 4.Exit \n");
while(1)
{
printf("enter your choice \n");
scanf("%d",&i);
switch(i)
{
case 1:if(ct==l)
printf("\n queue is full \n");
else
{
printf("enter the element \n");
scanf("%d",&item);
if(ct==0)
insert(item,0);
else
insert(item,l);
}
break;
case 2:if(ct==0)
{
printf("queue is empty \n");
break;
}
else
del();
break;
case 3:if(ct==0)
printf("\n queue is empty");
else
display();
break;
case 4:exit(1);
default:break;
}
}
}
void insert(int x,int ch)

```



```

{
if(ch==0)
{
head=((struct link*)malloc(sizeof(struct link)));
head->item=x;
cur=head;
head->ptr=NULL;
}
else
{
while(cur->ptr!=NULL)
{
cur=cur->ptr;
}
temp=((struct link*)malloc(sizeof(struct link)));
temp->item=x;
temp->ptr=NULL;
cur->ptr=temp;
cur=temp;
}
ct++;
}
void del()
{
cur=head;
printf("\n the element deleted is %d",cur->item);
cur=cur->ptr;
head=cur;
ct--;
}
void display()
{
cur=head;
printf("\n the element in the queue are");
while(cur->ptr!=NULL)
{
printf("%d->",cur->item);
cur=cur->ptr;
}
printf("%d->NULL",cur->item);
}
OUTPUT:

```

Enter the size of the queue:3

The linked list implementation of a queue

1.insertion

2.deletion

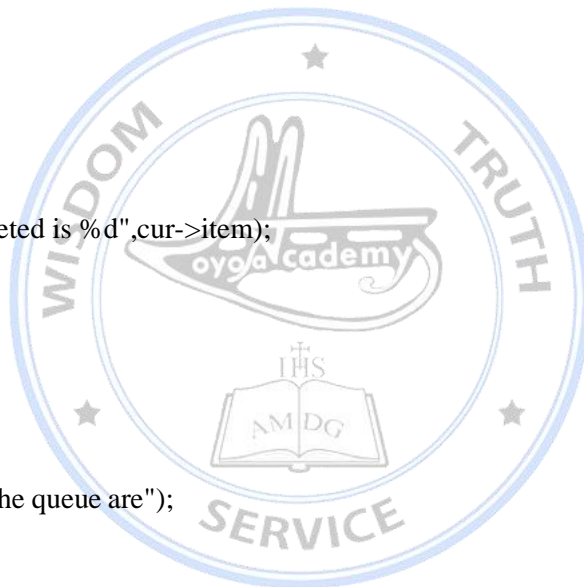
3.display

4.exit

Enter your choice:1

Enter the element 10

Enter your choice:1



Enter the element 20
Enter your choice:1
Enter the element 30
Enter your choice :3
10->20->30->NULL
Enter your choice :2
The deleted element is 10
Enter your choice :2
The deleted element is 20
Enter your choice :2
The deleted element is 30
Enter your choice :2
Queue is empty
Enter your choice :4

