DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

Academic Year: 2015-16 ODD SEMESTER

Programme (UG/PG)	**:**	**UG**
Semester	**:**	**IV**
Course Code	**:**	**CS1032**
Course Title	**:**	**Data Structures & Algorithms Lab**

Prepared By

R.ANNIE UTHRA
***AP(Sr.G), Department of Computer Science and Engineering*)**

FACULTY OF ENGINEERING AND TECHNOLOGY
SRM UNIVERSITY
(Under section 3 of UGC Act, 1956)
SRM Nagar, Kattankulathur- 603203
Kancheepuram District

EXERCISE NO-1(a) INSERTION SORT

Aim : To arrange the numbers in ascending order using insertion sort.

Algorithm:

1) Repeat step 2 to 5 for K=1 to n-1
2) Set temp=arr[k]
3) Set j=k-1
4) Repeat while temp <=arr[j]&&j>=0
   Set arr[j+1]=arr[j]
   Set j=j-1
5) Set arr [j+1] = temp
6) Exit

Program:

```
#include<stdio.h>
#include<conio.h>
voidinsertion_sort(int a[]);
void main()
{intarr[10],i;
printf("Enter the elements of the array");
for(i=0;i<10;i++)
  {
scanf("%d",&arr[i]);
  }
insertion_sort(arr);
getch();
}
voidinsertion_sort(int a[])
{
intk,j,temp,n;
  n=10;
for(k=1;k<=(n-1);k++)
  {
temp=a[k];
    j=k-1;
while(temp<=a[j] && j>=0)
    {
a[j+1]=a[j];
      j=j-1;
    }
a[j+1]=temp;
  }
printf("sorted array is:");
for(k=0;k<10;k++)
{  printf("%d\n",a[k]);
}
```

Output :

```
Enter the array
23
65
23
98
11
90
56
79
83
47
array11
23
23
47
56
65
79
83
90
98
```

Result : The elements were successfully arranged in ascending order using insertion sort

EXERCISE NO-1(b) SELECTION SORT

Aim : To arrange the numbers in ascending order using selection sort.

Algorithm:

1) Repeat step 2 and 3 for K=1 to n-1
2) Call smallest (arr ,k,n,pos)
3) Swap a[k] with arr[pos]
4) Exit

Smallest(arr,k,n,pos)

1) Set small = arr[k]
2) Set pos=k
3) Repeat for j=k+1 to n-1
   If small>arr[j]
   Set pos=j
   [end of if]
   [end of loop]
4) Exit

Program:

```
#include<stdio.h>
#include<conio.h>
int smallest(intarr[],intk,int n)
{
intsmall,j,pos;
small=arr[k];
pos=k;
for(j=k+1;j<n;j++)
  {
if(small>arr[j])
    {
small=arr[j];
pos=j;}}
returnpos;
}
void main()
{
inti,arr[6],k,c,temp;
clrscr();
printf("Enter 6 numbers: ");
for(i=0;i<6;i++)
{
scanf("%d",&arr[i]);
}
for(i=0;i<6;i++)
{
c=smallest(arr,i,6);
```

```
temp=arr[i];
arr[i]=arr[c];
arr[c]=temp;
}
for(i=0;i<6;i++)
{
printf("%d ",arr[i]);}
getch();
}
```

Output :

Result : The elements were successfully arranged in ascending order using selection sort.

EXERCISE NO-2 MERGE SORT

Aim : To sort the given elements using Merge sort.

Algorithm:

Merge _Sort(arr,beg,end)

1) If beg <end then

   Set mid =(beg + end)/2

Call Merge _Sort(arr,beg,mid)

   Call Merge _Sort(arr,mid+1,end)

   Call Merge _Sort(arr,beg,mid,end)

[end of if]

   Merge (arr,beg,mid,end)

1) Initialise set i = beg ; j=mid+1 ; index =0
2) Repeat while (i<=mid) and (j<=end)
   If arr[i]<arr[j],then
   Set temp[index] = arr[i]
   Set i=i+1
   else
   Set temp[index]=arr[j]
   Set j=j+1

[End of if]

Set index =index +1

[End of loop]

   Merge (arr,beg,mid,end)

3) [Copy the remaining elements of right sub array if any]
   If i> mid then
   Repeat while j <= end

Set index = index +1
[end of loop]
[copy the remaining elements of left sub. Array ,if any]
else
repeat while i<= mid
set temp[index]=arr[i]
set index = index + 1,sort i = i+1
[end of loop]

Program:

```
#include<stdio.h>
#include<conio.h>
intarr[10];
voidmerge_sort(intarr[],int beg, int end);
void merge(int a[],intbeg,intmid,int end);
void main()
{
inti;
clrscr();
printf("Enter the array");
for(i=0;i<10;i++)
  {
scanf("%d",&arr[i]);
  }
merge_sort(arr,0,9);
printf("array");
for(i=0;i<10;i++)
  {
            printf("%d",&arr[i]);
  }
getch();
}
voidmerge_sort(intarr[],intbeg,int end)
{inti;
if(beg<end)
{int mid=(beg+end)/2;
merge_sort(arr,beg,mid);
merge_sort(arr,mid+1,end);
merge(arr,beg,mid,end);
  }}
void merge(intarr[],intbeg,intmid,int end)
{inti,j,index,temp[10];
i=beg;
  j=mid+1;
index=beg;
while(i<=mid && j<=end)
  {if(arr[i]<arr[j])
    {temp[index]=arr[i];
i=i+1;
    }
else
    {temp[index]=arr[j];
    j=j+1;
    }
index=index+1;
  }
if(i>mid)
  {    while(j<=end)
{  temp[index]=arr[j];
index=index+1;
      j=j+1;
}}
else
```

```
{     while(i<=mid)
     {temp[index]=arr[i];
          index=index+1;
          i=i+1;
}}
for(i=beg;i<index;i++)
  {        arr[i]=temp[i];
  }}
```

Output :

Result : Hence the program for Sorting an array using merge sort was successfully executed.

EXERCISE NO-2(b) QUICK SORT

Aim : To sort the given elements using Quick sort.

Algorithm:

Quick _Sort(arr,beg,end)

1) If beg <end then

   Call partition(arr,beg,end,loc)

   Call Quick_Sort(arr,beg,mid+1)

   Call Quick_Sort(arr,loc+1,end)

[end of if]

2) end

partition (arr,beg,end,loc)

1) Initially set left = beg ;,right =end
   Loc=beg,flag = 0
2) Repeat steps 3 to 6 while flag = 0
3) Repeat while arr[loc]<=arr[right] and loc!=right
   Set right = right -1
   [End of loop]
4) If loc == right then
   Set flag=1
   else if arr[loc]>arr[right]
   then
   swaparr[loc] with arr[right]
   setloc = right

   [END OF IF**]**
5) If flag =0 then
   Repeat while arr[loc]>=arr[left]
   andloc!=left
   set left = left + 1

   [END OF LOOP**]**

6) If loc== left then
   Set flag =1
   Else ifarr[loc]<arr[left] then
   Swap arr[loc] with arr[left]
   Set loc = left
   [END OF IF**]**

[END OF LOOP**]**

7) end

Program:

```
#include<stdio.h>
#include<conio.h>
int part(int a[],intbeg,intend,intloc);
void quicksort(int a[],intbeg,int end);
int a[10];
void main()
{inti;
printf("enter the array");
for(i=0;i<10;i++)
{  scanf("%d",&a[i]);
  }
quicksort(a,0,9);
printf("sorted");
for(i=0;i<10;i++)
{   printf("%d\n",a[i]);
  }
getch();
}
void quicksort(int a[],intbeg,int end)
{     intloc=beg;
if(beg<end)
  {if(loc==beg)
   {loc= part(a,beg,end,loc);
   }
quicksort(a,beg,loc-1);
quicksort(a,loc+1,end);
  }}
int part(int a[],intbeg,intend,intloc)
{intleft,temp,right,flag,i;
left=beg;
right=end;
loc=beg;
flag=0;
while(flag==0)
 {   while((a[loc]<=a[right]) &&loc!=right)
   {     right=right-1;
   }
if(loc==right)
   {     flag=1;
   }
else if(a[loc]>a[right])
   {     temp= a[loc];
a[loc]=a[right];
a[right]=temp;
loc=right;
   }
if(flag==0)
   {     while((a[loc]>=a[left])  &&loc!=left)
     {       left=left+1;
}}
```

```
if(loc==left)
   {     flag=1;
   }
else if(a[loc]<a[left])
   {     temp=a[loc];
a[loc]=a[left];
a[left]=temp;
loc=left;
   }}
returnloc;
}
```

Output :

Result : Hence the program for Sorting an array using quick sort was successfully executed.

EXERCISE NO-3 STACKS

Aim : To write a menu driven program to perform following operations on the stack-

(i)Push (ii)Pop (iii) Peek

Algorithm:

Push-

1) If top=MAX-1 then print “overflow”
2) Set top=top+1;
3) Set stack[top]=val;
4) end

Pop-

1) if top==NULL then print “underflow”
2) set val=stack[top]
3) set top=top-1;
4) end

Peep-

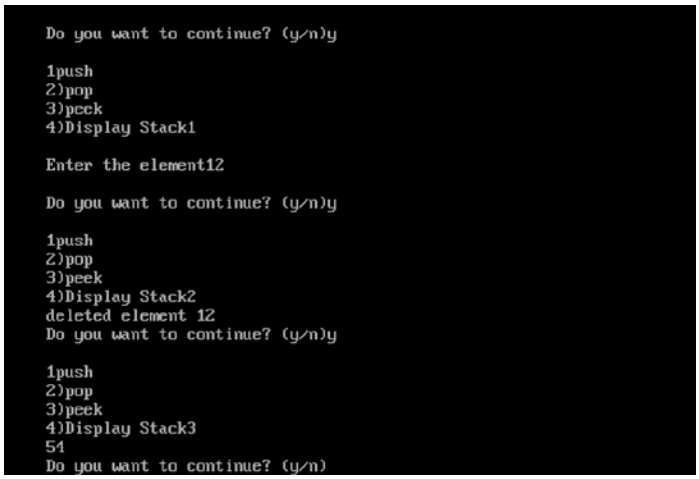
1) if top==NULL then print “empty stack”
   go to step 3
2) return stack[top];
3) end

Program:

```
#include<stdio.h>
#include<conio.h>
int a[5];int max=4;int top=-1;
void pop();void push();
void peek();  voiddisp();
void main()
{  intj,k;
charch;
clrscr();
do
{    printf("\n1push\n2)pop\n3)peek\n4)Display Stack") ;
scanf("%d",&j) ;
switch(j)
{
case 1: push();break;
case 2: pop();break;
case 3: peek();break;
case 4: disp();break;
default: printf("\ninvalid choice");   }
printf("\nDo you want to continue? (y/n)");
scanf("%s",&ch);
}while(ch=='y');
getch();   }
void push()
{   intval;
```

```
if(top==max)
printf("\nstack overflow\n");
else
  { top++;
printf("\nEnter the element");
scanf("%d",&val);
a[top]=val;}}
void pop()
{   intval;
if(top==-1)
printf("\nstack underflow\n");
else
  {   val=a[top];
top--;
printf("deleted element %d", val);
}}
void peek(int a[],int top)
{    if(top==-1)
printf("empty stack\n");
else
printf("%d",a[top]);
}
voiddisp()
{   printf("Elemnts of stack are\n");
            for(k=0;k<=top;k++)
            {
            printf("%d",a[k]);
             }}
```

Output :



Result : Hence the program forimplementing stack operations like push,pop and peek were successfully executed.

EXPERIMENT NO. 4 LIST USING ARRAYS

AIM: To perform list operations with the help of arrays using C programming

ALGORITHM

main()

1. Display the operation that can be performed
2. Repeat step 3 and 4 uptil x!=7
3. x← choice of user
4. switch x
   case 1: insertion(y,p)
   case 2: deletion()
   case 3: findpos()
   case 4: index(y)
   case 5: empty()
   case 6: display()
   [end switch]
   [[end do while]

insertion(int a, int b)

1. If end>9
   print overflow
2. Else
   For I from end +1 to b
   Arr(i+1)←Arr(i)

deletion(int a, int b)

1. For I in range a to end
   Arr(i)←Arr(i+1)
2. (end for)
3. end←end+1

findpos(int a)

1. if a>=0 and a<=end
   Print Arr(a)
2. Else
   Print position doesn't exist

Empty()

1. end← -1

display()

1. for i=0 to end
   Print Arr(i)

Program-

```
#include<stdio.h>
void insert();
void delete();
voidmakeempty();
voidfindpos();
voidfindele();
voiddisp();
intarr[100],size=0;
int main()
{
        intch;
        do
        {
                printf("\n1.Insertion");
                printf("\n2.Deletion");
                printf("\n3.Make Empty");
                printf("\n4.Find element");
                printf("\n5.Find element pos");
                printf("\n6.Display");
                printf("\n7.Exit");
                printf("\nEnter choice:-");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: insert();
                                        break;
                        case 2: delete();
```

```
                                        break;
                        case 3: makeempty();
                                        break;
                        case 4: findpos();
                                        break;
                        case 5: findele();
                                        break;
                        case 6: disp();
                                        break;
                }
        }while(ch<=6);
        return 0;
}

void insert()
{
        if(size==100)
        {
                printf("\nOverflow");
        }
        else
        {
                inti,no,pos;
                printf("\nEnter position :-");
                scanf("%d",&pos);
                if(pos<=size)
                {
                        printf("Enter elelment:-");
```

```
                scanf("%d",&no);
                for(i=size;i>=pos;i--)
                {
                        arr[i+1]=arr[i];
                }
                arr[pos]=no;
                ++size;
                printf("\nElement %d inserted\n",no);
        }
        else
        {
                printf("\nInvalid position");
        }
    }
}

void delete()
{
    intpos,i;
    if(size==0)
    {
        printf("Underflow\n");
    }
    else
    {
        printf("\nEnter position :-");
        scanf("%d",&pos);
        if(pos<=size)
```

```
                {
                        for(i=pos;i<size;i++)
                        {
                                arr[i]=arr[i+1];
                        }
                        --size;
                        printf("\nElement deleted");
                }
                else
                {
                        printf("\nInvalidpostion");
                }
        }
}

voidmakeempty()
{
        size=0;
        printf("\nArray Emptied");
}

voidfindpos()
{
        intpos;
        printf("\nEnter position :");
        scanf("%d",&pos);
        if(pos<=size)
        {
```

```
		printf("\nElement at postion %d is : %d",pos,arr[pos]);
	}
	else
	{
		printf("\n Invalid Position");
	}
}

voidfindele()
{
	inti,no,pos=-1;
	printf("Enter element : ");
	scanf("%d",&no);
	for(i=0;i<size;i++)
	{
		if(arr[i]==no)
		{
			pos=i;
			break;
		}
	}
	if(pos!=-1)
	{
		printf("\nElement %d found at %d position",no,pos);
	}
	else
	{
		printf("\nElement not found");
```

```
        }
}

voiddisp()
{
        inti;
        printf("\n List is :-\n");
        for(i=0;i<size;i++)
        {
                printf("%d\t",arr[i]);
        }
}
```

EXERCISE NO-5 QUEUES

Aim : To perform queue operations on the menu-

(i)Insert (ii)Delete (iii)Peek (iv)Display

Algorithm:

Insert-

1) If rear=MAX-1 then print “overflow”
2) If front==-1 and rear=-1 then set front=rear=0
   Else
   Set rear=rear+1;
3) Set queue[rear]num
4) end

Delete-

1) if front=-1 or front>rear then print “underflow”
   else
   set front=front+1;
   setval=queue[front];
2) end
3)

Peek-

1) Print queue[front]
2) end

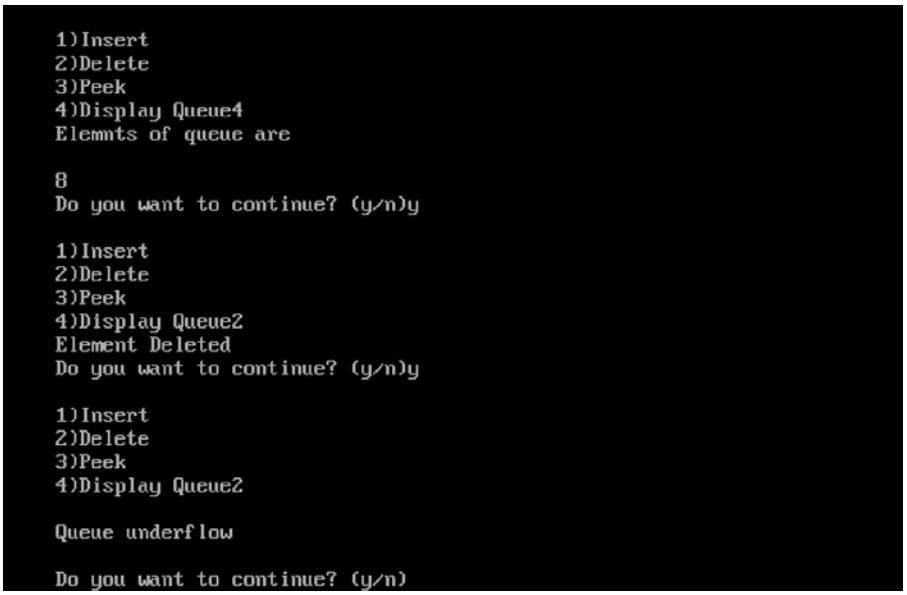
Program:

```
#include<stdio.h>
#include<conio.h>
int a[5];   int rear=-1;
int front=-1;int max=5;
void delete();   void insert();void peek();
void main()
{intj,k;
charch;
clrscr();
do{
printf("\n1)Insert\n2)Delete\n3)Peek\n4)Display Queue") ;
scanf("%d",&j) ;
switch(j)
{  case 1: insert();break;
```

```
case 2: delete();break;
case 3: peek();break;
case 4: printf("Elemnts of queue are\n");
        for(k=front;k<=rear;k++)
        {       printf("\n%d",a[k]);
         }
        break;
default: printf("\ninvalid choice");
 }
printf("\nDo you want to continue? (y/n)");
scanf("%s",&ch);
}while(ch=='y');
getch();
}
void peek()
{   printf("Peek element: %d",a[front]);
}

void insert()
{   intval;
printf("\nEnter the element");
scanf("%d",&val);
if(rear==max-1)
printf("\nQueue overflow\n");
if (front==-1 && rear==-1)
front =rear=0;
else
rear++;
a[rear]=val;   }
void pop()
{   intval;
if(front==-1||front>rear)
printf("\nQueue underflow  \n");
else
  {   front++;
```

```
val=a[front];
printf("Element Deleted");
  }}
```

Output :



Result : Hence the program forqueue operations was successfully executed.

EXERCISE NO-6 SINGLY LINKED LIST

Aim : To write program to create linked list and perform the following function

a) Insertion
b) Deletion
c) Searching
d) Display

Algorithm:

Insertion at beginning

1) Allocate memory for new node.
2) Set new_node→data = val
3) Set new_node→next = start
4) Set start = new _node
5) End.

Insertion at end

3) Set new_node→ next = null
4) Set ptr = start
5) Set step 6 while
   Ptr→ next!=NULL
6) Set ptr = ptr→ next

   [END OF LOOP**]**
7) Set ptr→ next = new_node
8) Exit

Search

1) Initialise set ptr = start
2) Repeat step 3 while (ptr)!=NULL
3) If val = ptr→ data
   Set pos = ptr
   Gotostep 5
4) Set pos = NULL(-1)
5) Exit

Delete

1) Set ptr = Start
2) Set start = start → next
3) Free ptr
4) exit

Program:

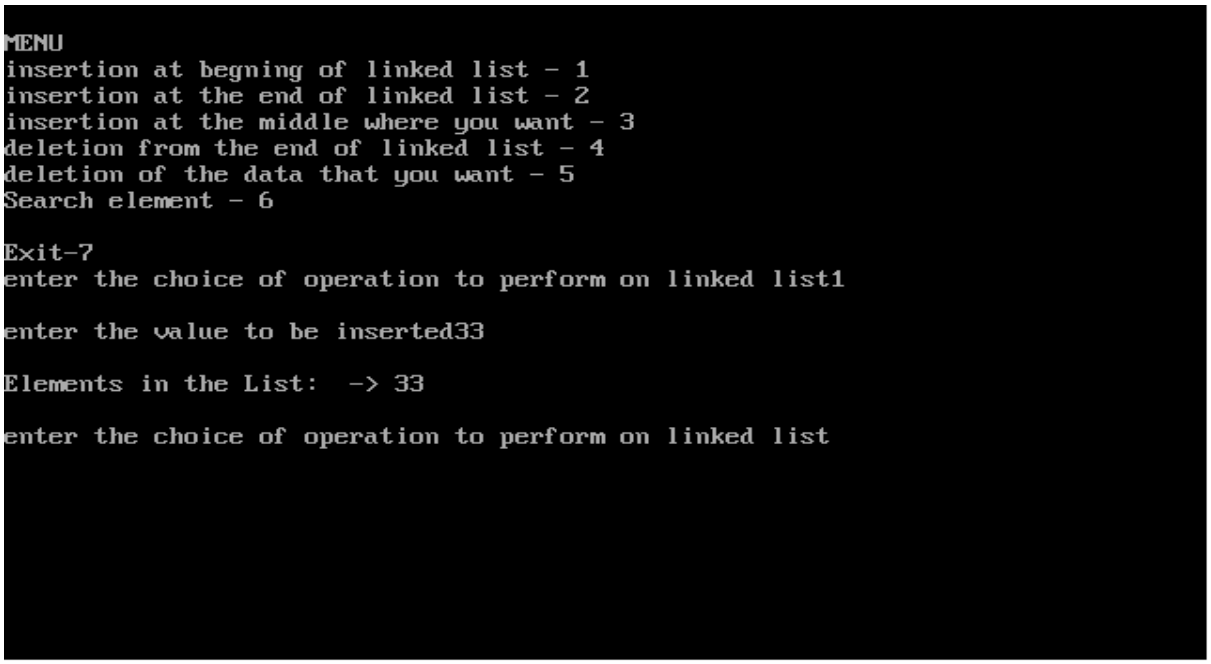
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{    int data;
struct node *next;
}*head,*var,*trav;

voidinsert_at_begning(int value)
{     var=(struct node *)malloc(sizeof (struct node));
var->data=value;
if(head==NULL)
    {        head=var;
head->next=NULL;
    }
else
    {        var->next=head;
head=var;
    }}
voidinsert_at_end(int value)
{      struct node *temp;
temp=head;
var=(struct node *)malloc(sizeof (struct node));
var->data=value;
if(head==NULL)
     {         head=var;
head->next=NULL;
     }
else
     {         while(temp->next!=NULL)
        {    temp=temp->next;
        }
var->next=NULL;
temp->next=var;
     }}
voidinsert_at_middle(int value, intloc)
{     struct node *var2,*temp;
var=(struct node *)malloc(sizeof (struct node));
var->data=value;
temp=head;
if(head==NULL)
    {         head=var;
head->next=NULL;     }
else
    {         while(temp->data!=loc)
        {              temp=temp->next;}
            var2=temp->next;
          temp->next=var;
          var->next=var2;
    }}
intdelete_from_middle(int value)
{     struct node *temp,*var;
temp=head;
```

```
while(temp!=NULL)
    {      if(temp->data == value)
             {                    if(temp==head)
                          {          head=temp->next;
                          free(temp);
                          return 0;
                          }
                          else
                          {var->next=temp->next;
                          free(temp);
                          return 0;
                          }}
             else
               {        var=temp;
             temp=temp->next;
               }}
printf("data deleted from list is %d",value);
return 0;}
intdelete_from_end()
{     struct node *temp;
temp=head;
while(temp->next != NULL)
    {         var=temp;
temp=temp->next;     }
if(temp ==head)
    {         head=temp->next;
free(temp);
return 0;     }
printf("data deleted from list is %d",temp->data);
var->next=NULL;
free(temp);
return 0;}
void display()
{     trav=head;
if(trav==NULL)
    {      printf("\nList is Empty");
    }
else
    {      printf("\nElements in the List: ");
           while(trav!=NULL)
           {printf(" -> %d ",trav->data);
           trav=trav->next;           }
printf("\n");
     }}
void search(int value)
{struct node *ptr;
for(ptr=head;ptr!=NULL;ptr=ptr->next)
{if(ptr->data==value)
{printf("Key found");
printf(ptr);
return;
}}
printf("\n %d not found",value);
}
void main()
```

```
{
inti=0;
head=NULL;
printf("\ninsertion at begning of linked list - 1");
printf("\ninsertion at the end of linked list - 2");
printf("\ninsertion at the middle where you want - 3");
printf("\ndeletion from the end of linked list - 4");
printf("\ndeletion of the data that you want - 5");
printf("\nSearch element - 6\n");
printf("\nExit-7");
while(1)
    {   printf("\nenter the choice of operation to perform on linked list");
        scanf("%d",&i);
        switch(i)
          {             case 1:
                {       int value;
                printf("\nenter the value to be inserted");
                scanf("%d",&value);
                insert_at_begning(value);
                display();
                break;}
                case 2:
                {       int value;
                printf("\nenter value to be inserted");
                scanf("%d",&value);
                insert_at_end(value);
                display();
                break;          }
                case 3:
                {       intvalue,loc;
                printf("\nafter which data you want to insert the data");
                scanf("%d",&loc);
                printf("\nenter the value to be inserted");
                scanf("%d",&value);
                insert_at_middle(value,loc);
                display();
                break;}
                case 4:
                {delete_from_end();
                display();
                break;}
                case 5:
                {int value;
                display();
                printf("\nenter the data that you want to delete from the list shown above");
                scanf("%d",&value);
                delete_from_middle(value);
                display();
                break;}
                case 6:
                {   int value;
                printf("|nEnter the number you want t search");
                scanf("%d",&value);
                search(value);
                break;
```

```
                case 7:
                {exit(0);
                }} }}
getch();
}
```

Output :



Result : The operations on a linked list are successfully performed

EXERCISE NO-7 DOUBLY LINKED LIST

Aim : To write program to create Doubly linked list and perform the following function

a) Insert
b) Delete
c) Display
d) search

Algorithm:

Insert

1) Allocate memory for new_ node.
2) Set new_node→data = val
3) Set new_node→next = start
4) Set start = new _node
5) End.

Search

a) Initialise set ptr = start
b) Repeat step 3 while (ptr)!=NULL
c) If val = ptr→ data
Set pos = ptr
Gotostep 5
d) Set pos = NULL(-1)
e) Exit

Delete

1) Set ptr = Start
2) Set start = start → next
3) Free ptr
4) exit

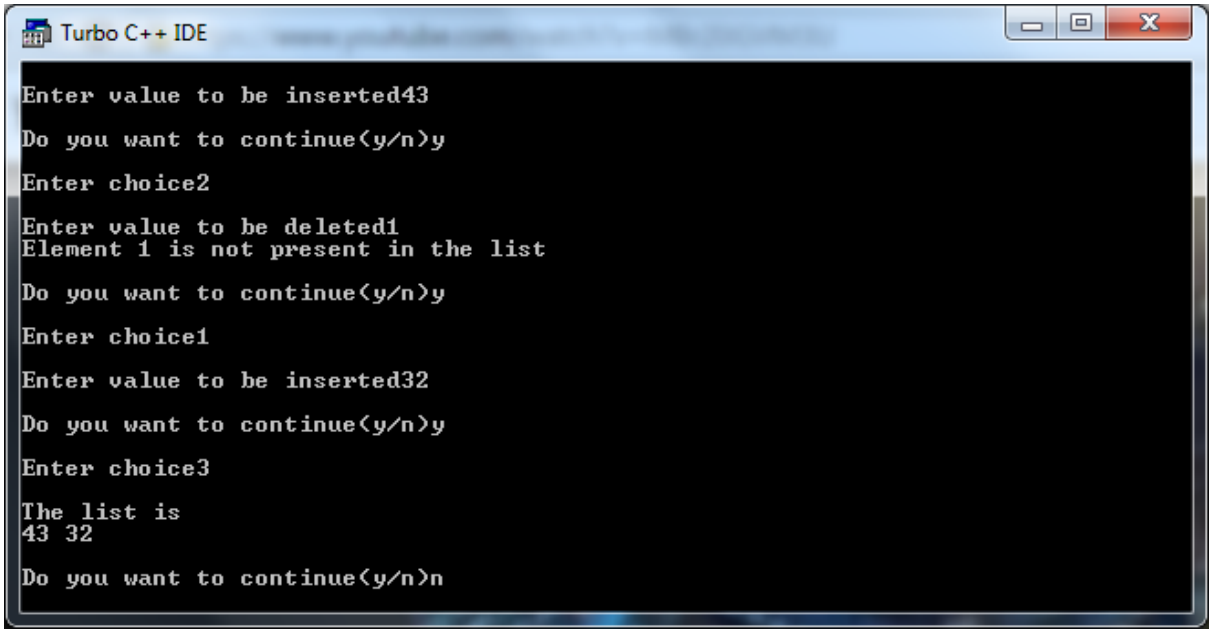
Program:

```
#include<stdio.h>
#include<stdlib.h>
typedefstruct Node
{
int data;
struct Node *next;
struct Node *prev;
}node;
void insert(node *pointer, int data)
{

while(pointer->next!=NULL)
```

```
        {
pointer = pointer -> next;
        }
pointer->next = (node *)malloc(sizeof(node));
        (pointer->next)->prev = pointer;
pointer = pointer->next;
pointer->data = data;
pointer->next = NULL;
}
int find(node *pointer, int key)
{
pointer =  pointer -> next;
while(pointer!=NULL)
        {
if(pointer->data == key)
                        {
                                return 1;
                        }
                        pointer = pointer -> next;
                }
                return 0;
}
void delete(node *pointer, int data)
{
node *temp;
                while(pointer->next!=NULL && (pointer->next)->data != data)
                {
                        pointer = pointer -> next;
                }
                if(pointer->next==NULL)
                {
                        printf("Element %d is not present in the list\n",data);
                        return;
                }
                temp = pointer -> next;
                pointer->next = temp->next;
                temp->prev =  pointer;
                free(temp);
                return;
}
void print(node *pointer)
{
                if(pointer==NULL)
                {
                        return;
                }
                printf("%d ",pointer->data);
                print(pointer->next);
}
void  main()
{
                node *start,*temp;
                charch;
                intquery,data;
                clrscr();
```

```
        start = (node *)malloc(sizeof(node));
        temp = start;
        temp -> next = NULL;
        temp ->prev = NULL;
        printf("1.  Insert\n");
        printf("2.  Delete\n");
        printf("3.  Print\n");
        printf("4.  Find\n");

do
{
printf("\nEnter choice");
                scanf("%d",&query);
                switch(query)
                {
                        case 1:
                        printf("\nEnter value to be inserted");
                        scanf("%d",&data);
                        insert(start,data);
                        break;
                        case 2:
                        printf("\nEnter value to be deleted");
                        scanf("%d",&data);
                        delete(start,data);
                        break;
                        case 3:
                        printf("\nThe list is ");
                        printf("\n");
                        print(start->next);
                        printf("\n");
                        break;
                        case 4:
                        printf("\nEnter value to be searched");
                        scanf("%d",&data);
                        if(find(start,data))
                        {
                                printf("Element Found\n");
                        }
                        else
                        {
                                printf("Element Not Found\n");
                        }
                        break;
                        default:
                        printf("\nWrong choice");
                        break;
                        }
                        printf("\nDo you want to continue(y/n)");
                        scanf("%s",&ch);
                        }while(ch=='y');
getch();
}
```

Output :



Result : The operations on a Doubly linked list are successfully performed

EXERCISE NO-8 TREE TRAVERSAL

Aim : To implement binary search tree traversal.

Algorithm:

preorder(node)

1) repeat step 2 to step 4 while tree!=NULL
2) write "TREE→data"
3) preorder(Tree→left)
4) Preorder(Tree→Right)
5) End

postorder(node)

1) repeat step 2 to step 4 while tree!=NULL
2) postorder(Tree→left)
3) Postorder(Tree→Right)
4) Write"TREE→data"
5) End

inorder(node)

1) repeat step 2 to step 4 while tree!=NULL
2) inorder(Tree→left)
3) write "TREE→data"
4) inorder(Tree→Right)
5) End

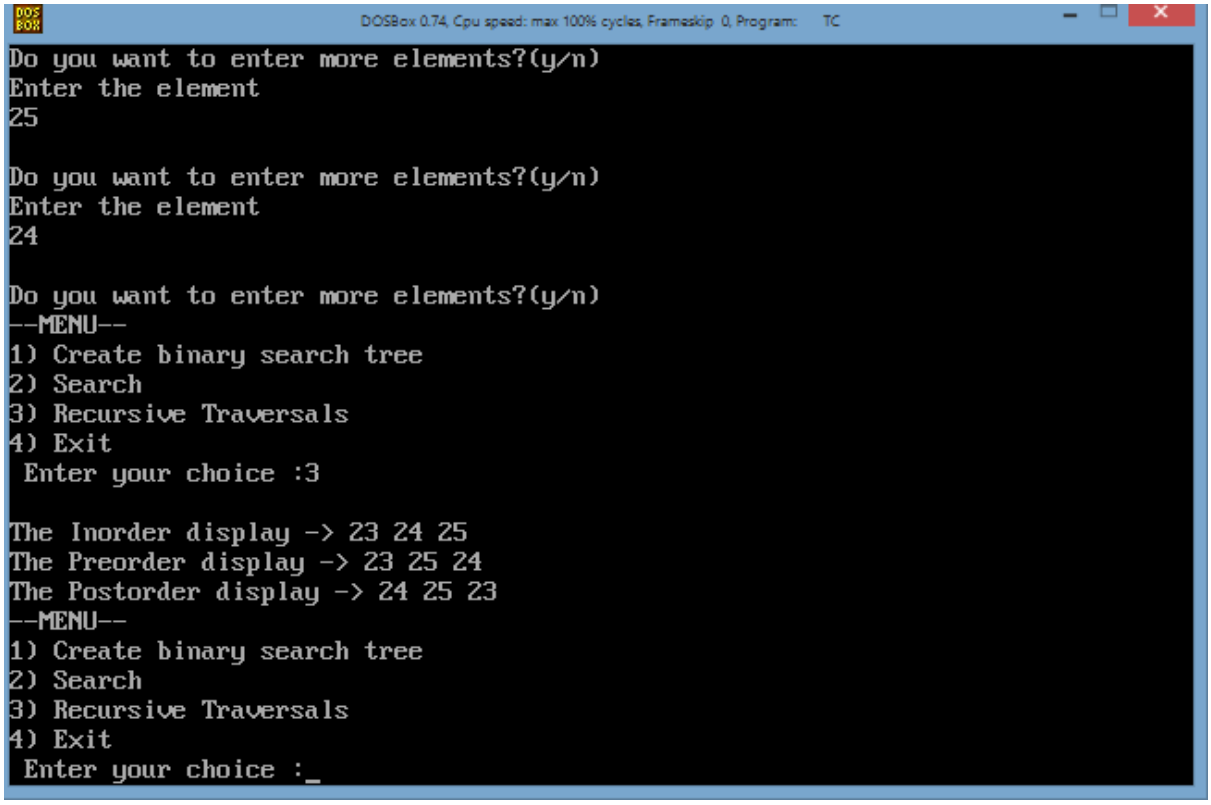
Program:

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

typedefstruct BST {
int data;
struct BST *lchild, *rchild;
} node;
void insert(node *, node *);
voidinorder(node *);
voidpreorder(node *);
voidpostorder(node *);
node *search(node *, int, node **);

void main() {
int choice;
charans = 'N';
int key;
node *new_node, *root, *tmp, *parent;
node *get_node();
root = NULL;
clrscr();
printf("\nProgram For Binary Search Tree ");
do {
printf("\n--MENU--");
printf("\n1) Create binary search tree");
printf("\n2) Search");
```

```
printf("\n3) Recursive Traversals");
printf("\n4) Exit");
printf("\n Enter your choice :");
scanf("%d", &choice);
switch (choice) {
case 1:
        do {
        new_node = get_node();
        printf("\nEnter the element ");
        scanf("%d", &new_node->data);
if (root == NULL) /* Tree is not Created */
        root = new_node;
        else
        insert(root, new_node);
        printf("\nDo you want to enter more elements?(y/n)");
        ans = getch();
         } while (ans == 'y');
        break;
case 2:
        printf("\nEnter the element to be searched :");
        scanf("%d", &key);
        tmp = search(root, key, &parent);
        printf("\nParent of the node %d is %d", tmp->data, parent->data);
        break;
case 3:
        if (root == NULL)
        printf("Tree is not created");
        else {
        printf("\nTheInorder display -> ");
        inorder(root);
        printf("\nThePreorder display -> ");
        preorder(root);
        printf("\nThePostorder display -> ");
        postorder(root);
         }
        break;
    }
  } while (choice != 4);
}
node *get_node() {
node *temp;
temp = (node *) malloc(sizeof(node));
temp->lchild = NULL;
temp->rchild = NULL;
return temp;
}
void insert(node *root, node *new_node) {
if (new_node->data < root->data) {
if (root->lchild == NULL)
        root->lchild = new_node;
else
        insert(root->lchild, new_node);
  }
if (new_node->data > root->data) {
if (root->rchild == NULL)
```

```
            root->rchild = new_node;
else
            insert(root->rchild, new_node);
   }}
node *search(node *root, int key, node **parent) {
node *temp;
temp = root;
while (temp != NULL) {
if (temp->data == key) {
            printf("\nThe %d Element is Present", temp->data);
            return temp;
      }
      *parent = temp;
if (temp->data > key)
            temp = temp->lchild;
else
            temp = temp->rchild;
   }
return NULL;
}
voidinorder(node *temp) {
if (temp != NULL) {
inorder(temp->lchild);
printf("%d ", temp->data);
inorder(temp->rchild);
   }}
voidpreorder(node *temp) {
if (temp != NULL) {
printf("%d ", temp->data);
preorder(temp->lchild);
preorder(temp->rchild);
   }}
voidpostorder(node *temp) {
if (temp != NULL) {
postorder(temp->lchild);
postorder(temp->rchild);
printf("%d ", temp->data);
   }}
```

Output :



Result : Hence the program forimplementing binary search tree traversal was successfully executed.

EXERCISE NO-9 GRAPH TRAVERSAL

Aim : To implement breadth first search (bfs) and depth first search (dfs) in graphs using adjacency matrix.

Algorithm:

bfs(ints,int n)

1) list L = empty
2) tree T = empty
3)choose a starting vertex x
4)search(x)
5) while(L nonempty)
6) remove edge (s,n) from start of L
7)if n not yet visited
add (s,n) to T
search(n)

dfs(ints,int n)

1) list L = empty
2) tree T = empty
3)choose a starting vertex x
4)search(x)
5) while(L nonempty)
6) remove edge (s,n) from end of L
7)if n not yet visited
add (s,n) to T
search(n)

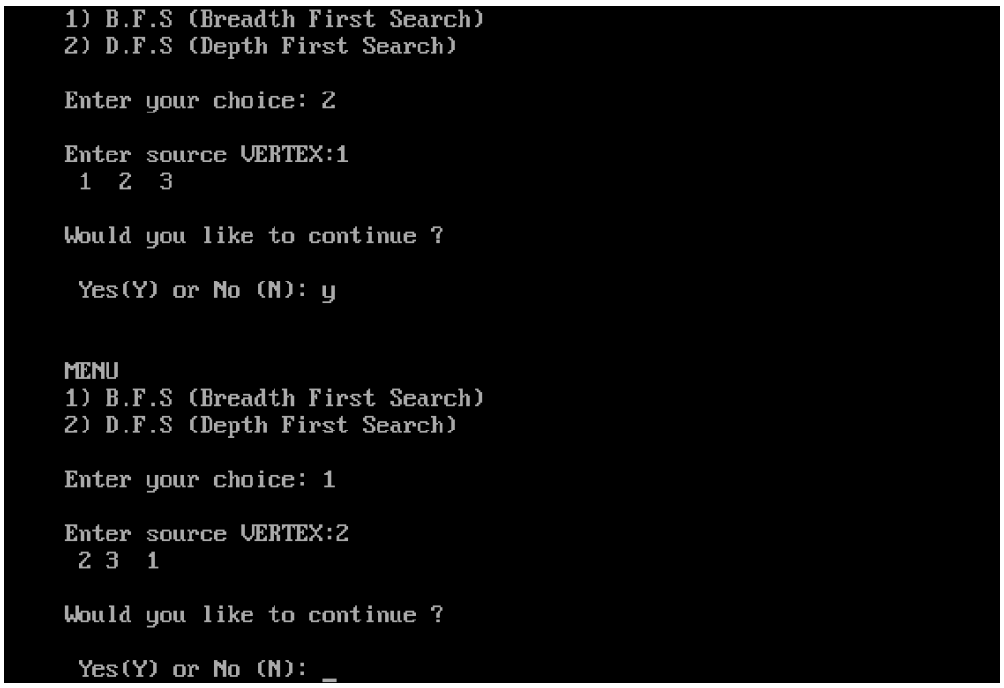
Program:

```
#include<stdio.h>
#include<conio.h>
int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
voidbfs(ints,int n);
voiddfs(ints,int n);
void push(int item);
int pop();
void main()
{  intn,i,s,ch,j;
charc,dummy;
clrscr();
printf("\n\n\n\t\t\tDFS and BFS implementation !");
printf("\n\n\nEnter the Number of Nodes in Graph: ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{    for(j=1;j<=n;j++)
{   printf("\nEnter 1 if %d has a NODE with %d else 0 ",i,j);
```

```
scanf("%d",&a[i][j]);
}}
printf("\n\n\t\tTheAdjecency Matrix is: ");
for(i=1;i<=n;i++)
{     for(j=1;j<=n;j++)
{   printf(" %d",a[i][j]);
}
printf("\n");
}  do
{  for(i=1;i<=n;i++)
vis[i]=0;
printf("\n\nMENU");
printf("\n1) B.F.S (Breadth First Search)");
printf("\n2) D.F.S (Depth First Search)");
printf("\n\nEnter your choice: ");
scanf("%d",&ch);
printf("\nEnter source VERTEX:");
scanf("%d",&s);
switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
} printf("\n\nWould you like to continue ?\n\n Yes(Y) or No (N): ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
} void bfs(ints,int n)
{   intp,i;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf(" %d",p);
while(p!=0)
{  for(i=1;i<=n;i++)
if((a[p][i]!=0)&&(vis[i]==0))
{   add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}
void add(int item)
{  if(rear==19)
printf("\n\nQUEUE is FULL: Error No Space!");
else
{   if(rear==-1)
```

```
{   q[++rear]=item;
front++;
}
else
q[++rear]=item;
}}
int delete()
{  int k;
if((front>rear)||(front==-1))
return(0);
else
{   k=q[front++];
return(k);
}}
voiddfs(ints,int n)
{   inti,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{  for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
{  push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{   if(top==19)
printf("\n\nStackoverflow.com");
else
stack[++top]=item;
}
int pop()
{  int k;
if(top==-1)
return(0);
else
{  k=stack[top--];
return(k);
}}
```

Output :



Result : Hence the program forimplementing breadth first search (bfs) and depth first search (dfs) in graphs using adjacency matrix was successfully executed.