LABORATORY MANNUAL
FOR
DATA STRUCTURE LAB
USING C

3rd Semester

Diploma in Computer Science & Engineering

Government Polytechnic,Nuapada

Kala Bhera, Nuapada, Odisha,PIN-766105

Prepared by
K. K. Chapeyar, Sr. Lecturer, CSE

Department of Computer Science & Engineering

Experiment1:

Implementation of 1D & 2D Array

Experiment1.1

Write a program to traverse an array using C.

Program:

```
#include<stdio.h>
void main()
{
  int a[6], i;
  printf("Enter the elements of the array :");
for(i=0;i<6;i++)
 {
        scanf("%d", &a[i]);
 }
printf("\n The elements of the array are:");
for(i=0; i<6; i++)
 {
        printf("\t%d",a[i]);
 }
}
```

OUTPUT:-

```
Enter the elements of the array:2
 3
 5
 7
 9
 6
The elements of the array are: 2       3       5       7       9       6
```

Experiment1.2

Write a program to insert an element to an array using C.

Program:

```
#include<stdio.h>
void main()
{
int a[50],i,n;
int x, pos;
printf("Enter the number of elements in the array \n");
scanf("%d",&n);
printf("Enter the elements of the array :");
for(i=0;i<n;i++)
 {
        scanf("%d", &a[i]);
 }
printf("\nThe entered elements of the array are:");
for(i=0;i<n; i++)
  {
        printf("\t%d",a[i]);
  }
printf("\n Enter the new element to be inserted:");
scanf("%d",&x);
printf("Enter the position where element is to be inserted: ");
scanf("%d",&pos);
for(i=n-1;i>=pos-1;i--)
   {
        a[i+1]=a[i];
    }
a[pos-1]=x;
 n=n+1;
printf("\nThe elements of the array after insertion:");
for(i=0;i<n;i++)
 {
printf("\t%d",a[i]);
 }

}
```

OUTPUT:-

```
Enter the number of elements in the array:6
Enter the elements of the array: 2  3  5  7 6  8
The entered elements of the array are: 2        3       5       7       6       8
Enter the new element to be inserted: 11
Enter the position where element is to be inserted:4
The elements of the array after insertion: 2    3       5       11      7       6       8
```

Experiment1.3

Write a program to delete an element from an array using C.

Program:

```
#include<stdio.h>
void main()
{
   int i, n, k,arr[10];
   printf("Enter the size of the array:");
   scanf("%d", &n);
   printf("Enter the elements of the array:\n");
   for (i = 0; i < n; i++)
   {
      printf("arr[%d]=", i);
      scanf("%d",&arr[i]);
   }
   printf("Enter the position of the element to be
   deleted:");
   scanf("%d", &k);
   if(k >n)
   {
      printf(" \n Deletion is not possible in the array.");
   }
   else
   {
      for(i=k-1;i<n-1;i++)
         arr[i] = arr[i + 1];
         printf("The array after deleting the element is:");
      for (i = 0; i < n - 1; i++)
         printf("%d\t",arr[i]);
   }
}
```

OUTPUT:-

```
Enter the size of the array: 5
Enter the elements of the array:
arr[0]=6
arr[1]=5
arr[2]=4
arr[3]=7
arr[4]=2
Enter the position of the element to be deleted: 3
 The array after deleting the element is: 6      5      7     2
```

Experiment1.4

Write a program to sum two matrix by implementing 2D array using C.

Program:

```
#include <stdio.h>
void main()
{
int r,c,a[50][50], b[50][50],sum[50][50],i,j;
printf("Enter the number of rows (between 1 and 50):");
scanf("%d",&r);
printf("Enter the number of columns (between 1 and 50):");
scanf("%d", &c);
printf("\nEnter elements of 1st matrix:\n");
for(i=0;i<r;i++)
        for(j=0;j<c;j++)
        {
                printf("Enter element a%d%d:",i+1,j+1);
                scanf("%d",&a[i][j]);
        }
printf("Enter elements of 2nd matrix:\n");
for(i=0;i<r;i++)
        for(j=0;j<c;j++)
        {
        printf("Enter element b%d%d: ", i+1,j+1);
        scanf("%d",&b[i][j]);
        }
for(i=0;i<r;i++)
        for(j=0;j<c;j++)
                {
                        sum[i][j] = a[i][j] + b[i][j];
                }
printf("\nSum of two matrices: \n");
for(i=0;i<r;i++)
        for(j=0;j<c;j++)
        {
        printf("%d ",sum[i][j]);
        if (j==c-1)
          {
                printf("\n");
          }
    }

}
```

OUTPUT:-

Enter the number of rows (between 1 and 50): 2
Enter the number of columns (between1and50): 3

Enter elements of 1st matrix:
Enter element a11: 2
Enter element a12:3
Enter element a13:4
Enter element a21:5
Enter element a22:2
Enter element a23:3
Enter elements of 2nd matrix:
Enter element b11: -4
Enter element b12:5
Enter element b13:3
Enter element b21:5
Enter element b22:6
Enter element b23:3

Sum of two matrices:
-2 8 7

10 8 6

Experiment2:-

Write a program to implement stack using C.

Program:

```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
void main()
{
top=-1;
printf("\n Enter the size of  STACK[MAX=100]:");
scanf("%d",&n);
printf("\n\t STACK OPERATIONS USING ARRAY");
printf("\n\t ----------------------------------------------");
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
do
{
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
                {
                push(); break;
                }
        case 2:
                {
                pop(); break;
                }
        case 3:
                {
                display(); break;
                }
        case 4:
                {
                printf("\n\t EXIT "); break;
                }
        default:
                {
                printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
                }

  }
}while(choice!=4);
}
void push()
{
        if(top>=n-1)
```

```
        {
        printf("\n\t STACK is OVERFLOW");
        }
        else
        {
        printf(" Enter a value to be pushed:");
        scanf("%d",&x);
        top++;
        stack[top]=x;
        }
}
void pop()
{
        if(top<=-1)
        {
        printf("\n\t Stack is UNDERFLOW");
        }
        else
        {
        printf("\n\t The popped element is: %d",stack[top]);
        top--;
        }
}
void display()
{
        if(top>=0)
        {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
                printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
        }
        else
        {
        printf("\n The STACK is empty");
        }
}
```

OUTPUT:-

```
Enter the size of STACK[MAX=100]:10

        STACK OPERATIONS USING ARRAY

        1. PUSH
        2. POP
        3. DISPLAY
        4. EXIT
 Enter the Choice:1
 Enter a value to be pushed:5
 Enter the Choice:1
 Enter a value to be pushed:6
 Enter the Choice:1
 Enter a value to be pushed:7
 Enter the Choice:3
 The elements in STACK
7
6
5
 Press Next Choice
 Enter the Choice:2
        The popped element is: 7
        Enter the Choice:3
 The elements in STACK
 6
5
 Press Next Choice
 Enter the Choice:4

        EXIT
```

Experiment3:

Write a program to implement pointer using C.

What is

a pointer in C?

A pointer is a derived data type that can store the address of other C variables or a memory location. We can access and manipulate the data stored in that memory location using pointers.

Program:

```
#include <stdio.h>
void main()
{
int var1=10;
int *ptr1;
int **ptr2;
ptr1=&var1;
ptr2 =&ptr1;
printf("Value at ptr1 = %p \n",ptr1);
printf("Value at var1=%d\n",var1);
printf("Value of variable using *ptr1=%d\n",*ptr1);
printf("Value at ptr2 = %p \n",ptr2);
printf("Value stored at *ptr2 = %d \n", *ptr2);
printf("Value of variable using **ptr2=%d\n",**ptr2);
}
```

OUTPUT:-

```
Value at ptr1 = 000000000062FE14
Value at var1 = 10
Value of variable using *ptr1 = 10
Value at ptr2 = 000000000062FE08
Value stored at *ptr2 = 6487572
Value of variable using **ptr2 = 10
```

Experiment4:

Structure & Union

Experiment4.1

Write a program to implement Structure using C.

Program:

```
#include<stdio.h>
struct EMP
{
   int id;
   char name[50];
   int sal;
} ;
void main()
{
   EMP te;
   printf("Enter EMP ID:");
   scanf ("%d", &te.id);
   printf("Enter EMP Name:");
   scanf ("%s", te.name);
   printf("Enter EMP salary:");
   scanf ("%d", &te.sal);

   printf("\n EMP ID:%d",te.id);
   printf("\n EMP Name:%s",te.name);
   printf("\n EMP Salary:%d",te.sal);
}
```

OUTPUT

```
Enter EMP ID:101
Enter EMP Name:Sagun
Enter EMP salary:2000

 EMP ID:101
 EMP Name:Sagun
 EMP Salary:2000
```

Experiment4.2

Write a program to implement Structure using C.

```
#include<stdio.h>
typedef struct
{
int id;
char name[40];
float salary;
} EMP;
EMP getdata ()
{
EMP te;
printf ("Enter EMP ID:");
scanf ("%d", &te.id);
printf ("Enter EMP name:");
scanf ("%s", te.name);
printf ("Enter EMP salary:");
scanf ("%f", &te.salary);
return te;
}
void showdata (EMP te)
{
printf("\nEMP ID : %d NAME:%s SALARY:%0.2f",te.id, te.name, te.salary);
}
float sumsal(float  s1, float  s2)
{
return (s1 + s2);
}
int main ()
{
EMP e1, e2;
float tsal;
e1=getdata();
```

```
e2=getdata();
showdata(e1);
showdata(e2);
tsal = sumsal(e1.salary,e2.salary);
printf ("\nSum Salary = %0.2f", tsal);
}
```

OUTPUT:-

```
Enter EMP ID:1002
Enter EMP name:kalia
Enter EMP salary:4000.5
Enter EMP ID:1003
Enter EMP name:Romio
Enter EMP salary:6000.8

EMP ID : 1002 NAME:kalia SALARY:4000.50
EMP ID : 1003 NAME:Romio SALARY:6000.80
Sum Salary = 10001.30
```

Experiment4.3

Write a program to implement Union using C.

Program:

```
#include<stdio.h>
int main()
{

   int i;
   struct employee_imfo
   {
      int employee_id;
       union
      {
         long int adhar_card_number;
         long int voter_id_card_number;
         char other_id [10];
      };
      char goverment_id;
   }a,b;

   a.employee_id = 8;
   a.goverment_id='a';
   a.adhar_card_number=868796;

   b.employee_id=7;
   b.goverment_id='v';
   b.voter_id_card_number=1234;

   printf("Employee a Information:\n");
   if(a.goverment_id=='a')
   {
      printf("GOVT.ID PROVIDED IS ADHAR CARD_ID\n");
   }
   else
   {
      printf("GOVT.ID PROVIDED IS VOTER ID CARD\n");
   }

   printf("a id:%d\n a'sgovernment provided id:%d",a.employee_id,a.adhar_card_number);
   printf("\n\n");

   if(b.goverment_id=='a')
   {
      printf("GOVT.ID PROVIDED IS ADHAR CARD_ID\n");
   }
```

```
    else
    {
        printf("GOVT.ID PROVIDED IS VOTER ID CARD\n");
    }

    printf("Employee b Information :\n");
    printf("b id:%d\n b's government provided id is:%d",
    b.employee_id,b.voter_id_card_number);
    printf("\n\n");

    return 0;

}
```

OUTPUT:-

```
Employee a Information:
GOVT. ID PROVIDED IS ADHAR CARD_ID
a id:8
a's government provided id:868796

GOVT.ID PROVIDED IS VOTER ID CARD
Employee b Information:
b id: 7
b's government provided id:1234
```

Experiment5:-

Write a program to implement push and pop operation on Stack using C.

Program:

```
#include<stdio.h>
#include<stdlib.h>
int top=-1,size,i;
int stack[50];
 void push();
 void pop();
 void show();
int main()
{
        int choice;
        printf("\n Enter the size(MAX SIZE 50) of the stack:");
        scanf("%d",&size);
        printf("\n Perform operations on the stack:");
        printf("\n\t1.PUSH \n\t2.POP\n\t3.SHOW\n\t4.EXIT");

while(1)
{
        printf("\nEnter your choice: ");
        scanf("%d",&choice);

  switch(choice)
 {
case 1:
        push(); break;
case 2:
        pop(); break;
case 3:
        show(); break;
case 4:
       printf("\nEXIT ");
        exit(0);

default:
        printf("\nInvalid choice!!");
 }
 }
}

void push()
{
        int x;

        if(top==size-1)
        {
        printf("\nOverflow!!");
        }
```

```
        else
        {
        printf("\nEnter the element to be added onto stack:");
        scanf("%d", &x);
        top = top + 1;
        stack[top]=x;
        }
}

void pop()
{
        if(top==-1)
        {
        printf("\nUnderflow!!");
        }
        else
        {
        printf("\nPopped element:%d",stack[top]);
        top = top - 1;
        }
}

void show()
{
        if(top==-1)
        {
        printf("\nStack is Empty");
        }
        else
        {
        printf("\nElements present in the stack:\n");
        for(i=top;i>=0;i--)
        printf("%d\n",stack[i]);
}

   }
```

OUTPUT:-

```
Enter the MAX SIZE of the stack:3

 Perform operations on the stack:
      1.PUSH
      2.POP
      3.SHOW
      4.EXIT
Enter your choice: 1

Enter the element to be added onto stack:4

Enter your choice: 1
```

Enter the element to be added onto stack:6

Enter your choice: 1

Enter the element to be added onto stack:8

Enter your choice: 3

Elements present in the stack:
8
6
4

Enter your choice: 2

Popped element:8
Enter your choice: 3

Elements present in the stack:
6
4

Enter your choice: 2

Popped element:6
Enter your choice: 3

Elements present in the stack:
4

Enter your choice: 2

Popped element:4
Enter your choice: 3

Stack is Empty
Enter your choice: 2

Underflow!!
Enter your choice: 4

EXIT

Experiment6:-

Write a program to implement insertion and deletion in Queue.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 5
void enqueue();
void dequeue();
void show();
int queue[SIZE];
int Rear = - 1;
int Front=-1;
int i;
 void main()
{
int ch;
printf("INSERTION & DELETION IN QUEUE\n");
printf("-----------------------------------------------\n");
printf("\t 1.Insertion Operation\n");
printf("\t 2.Deletion Operation\n");
printf("\t 3.Display the Queue\n");
printf("\t 4.Exit\n");
while(1)
{
printf("\nEnter your choice of operations:");
scanf("%d", &ch);
switch(ch)
{
case 1:
          enqueue(); break;
case 2:
          dequeue(); break;
case 3:
          show(); break;
case 4:
         printf("\nEXIT ");

          exit(0);
default:
          printf("\n Incorrect choice");
}
}
}
void enqueue()
{
int item;
if (Rear == SIZE - 1)
          printf("\n Overflow");
else
```

```
{
if(Rear==-1&&Front==-1)
        Rear=Front=0;
else
        Rear=Rear+1;
printf("\n Element to be inserted in the Queue:");
scanf("%d", &item);
queue[Rear]=item;
}
}

void dequeue()
{
if(Front ==-1||Rear==-1)
{
printf("\nUnderflow"); return ;
}
else
{
if(Front==Rear)
{
printf("\nElement deleted from the Queue:%d",queue[Front]);
        Front=-1;
}
else
{
printf("\nElement deleted from the Queue:%d",queue [Front]);
Front = Front + 1;
}
}
}

void show()
{
if(Front==-1&&Rear==-1)
        printf("\nEmpty Queue");
else
{
printf("\nQueue:");
for(i=Front;i<=Rear;i++)
printf("%d ", queue[i]);
printf("\n");
}
}
```

OUTPUT:-

```
INSERTION & DELETION IN QUEUE
-----------------------------------------------
    1.Insertion Operation
    2.Deletion Operation
    3.Display the Queue
    4.Exit
```

Enter your choice of operations:1

Element to be inserted in the Queue:4

Enter your choice of operations:1

Element to be inserted in the Queue:5

Enter your choice of operations:1

Element to be inserted in the Queue:6

Enter your choice of operations:1

Element to be inserted in the Queue:7

Enter your choice of operations:1

Element to be inserted in the Queue:9

Enter your choice of operations:1

Overflow
Enter your choice of operations:3

Queue:4 5 6 7 9

Enter your choice of operations:2

Element deleted from the Queue:4
Enter your choice of operations:3

Queue:5 6 7 9

Enter your choice of operations:2

Element deleted from the Queue:5
Enter your choice of operations:2

Element deleted from the Queue:6
Enter your choice of operations:2

Element deleted from the Queue:7
Enter your choice of operations:3

Queue:9

Enter your choice of operations:2

```
Element deleted from the Queue:9
Enter your choice of operations:2

Underflow
Enter your choice of operations:4
        EXIT
```

Experiment7:-

Write a program to implement insertion and deletion in Linked List using C.

Program:

```
#include<stdio.h>
#include<stdlib.h>
void create();
void display();
void insert_begin();
void insert_end();
void insert_pos();
void delete_begin();
void delete_end();
void delete_pos();
struct node* head = NULL;
int i;
struct node
{
int data;
struct node* next;
};
int main()
{
int choice;
printf("\n*****\n");
printf("0. Create\n");
printf("1. display\n");
printf("2. Insert Node at beginning\n");
printf("3. Insert Node in specific position\n");
printf("4. Insert Node at end of LinkedList\n");
printf("5. Delete Node at beginning\n");
printf("6. Delete Node at end\n");
printf("7. Delete Node at position\n");
printf("8. ** To exit **");
while(1)
{
printf("\n Enter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 0: create();
                break;
case 1: display();
                break;
```

```
case 2: insert_begin();
                break;
case 3: insert_pos();
                break;
case 4: insert_end();
                break;
case 5: delete_begin();
                break;
case 6: delete_end();
                break;
case 7: delete_pos();
                break;
case 8: exit(0);
        printf("**EXIT**");

                break;
default: printf("\n Wrong Choice");
                break;
}
}
}
void create()
{
struct node* temp;
temp = (struct node*)malloc(sizeof(struct node));
printf("Enter node data: ");
scanf("%d",&temp->data);
temp->next = NULL;
        if(head==NULL)
         {
                head = temp;
          }
else{
        struct node* ptr = head;
        while(ptr->next!=NULL)
        {
        ptr = ptr->next;
   }
        ptr->next = temp; //inserting at end of List
  }
}
void display()
{
        if(head==NULL)
   {
        printf("Linked List is Empty\n");
        return;
   }
printf("Linked List is: ");
struct node* ptr = head;
while(ptr!=NULL) // start from first node
{
```

```
        printf("%d ",ptr->data);
    ptr = ptr->next;
}
printf("\n");
}

void insert_begin()
{
struct node* temp;
temp = (struct node*)malloc(sizeof(struct node));
printf("Enter node data: ");
scanf("%d",&temp->data);
temp->next = NULL;
if(head==NULL)
{
        head = temp;
        return;
}
else
{
        temp->next = head; //point it to old head node
        head = temp; //point head to new first node
}
}

void insert_pos()
{
struct node* temp;
// creating a new node
temp = (struct node*)malloc(sizeof(struct node));
printf("Enter node data: ");
scanf("%d",&temp->data);
temp->next = NULL;
if(head==NULL) // if list empty we return
{
        head = temp;
return;
}
else
{

struct node* prev_ptr;
struct node* ptr = head;
int pos;
printf("Enter position: ");
scanf("%d",&pos);
for(i=1;i<pos;i++)
{
prev_ptr = ptr;
ptr = ptr->next;
}
temp->next = ptr;
```

```
prev_ptr->next = temp;
}
}
void insert_end()
{
struct node* temp;
temp = (struct node*)malloc(sizeof(struct node));
printf("Enter node data: ");
scanf("%d",&temp->data);
temp->next = NULL;
if(head==NULL)
{
head = temp; //if list is empty, we return
return;
}
else{
struct node* ptr = head;
while(ptr->next!=NULL)
{
ptr = ptr->next;
}
ptr->next = temp;
}
}
void delete_begin()
{
if(head==NULL) //if List is empty we return
{
        printf("Linked List is empty | Nothing to delete \n");
return;
}
else
{
        struct node* ptr = head;
head = head->next; // head node pointing to second node
free(ptr); // deleting prev head node
printf("Node Deleted \n");
}
}


void delete_end()
{
if(head==NULL) //if List is empty we return
{
printf("Linked List is empty | Nothing to delete \n");
return;
}
else if(head->next==NULL)
{
        struct node* ptr = head;
head = ptr->next;
```

```
free(ptr);
printf("Node Deleted \n");
}
else
{
struct node* ptr = head;
struct node* prev_ptr = NULL;
while(ptr->next!=NULL)// traverse till last but one node
{
prev_ptr = ptr;
ptr = ptr->next;
}
prev_ptr->next = NULL; // next field of last but one field is made as NULL
free(ptr); // deleting last node
printf("Node Deleted \n");
}
}
void delete_pos()
{
int pos;
printf("Enter node position to delete: ");
scanf("%d",&pos);
struct node* ptr=head;
if(head==NULL) //we return if List is empty
{
printf("Linked List is empty \n");
return;
}
else if(pos==1)
{
        ptr = head;
head=ptr->next; // head pointing to second node
free(ptr); // deleting old first node
printf("Node Deleted \n");
}
else
{

struct node* prev_ptr;
for(i=1;i<pos;i++)
{
prev_ptr = ptr;
ptr = ptr->next;
}
prev_ptr->next = ptr->next; //prev node pointing to pos+1 node
free(ptr); //deleting node at pos
printf("Node Deleted \n");
}
}
```

OUTPUT:-

```
*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 1
Linked List is Empty

 Enter your choice: 0
Enter node data: 25

 Enter your choice: 0
Enter node data: 50

 Enter your choice: 1
Linked List is: 25 50

 Enter your choice: 2
Enter node data: 100

 Enter your choice: 1
Linked List is: 100 25 50

 Enter your choice: 4
Enter node data: 45

 Enter your choice: 1
Linked List is: 100 25 50 45

 Enter your choice: 3
Enter node data: 75
Enter position: 2

 Enter your choice: 1
Linked List is: 100 75 25 50 45

 Enter your choice: 5
Node Deleted

 Enter your choice: 1
Linked List is: 75 25 50 45

 Enter your choice: 6
```

Node Deleted

Enter your choice: 1
Linked List is: 75 25 50

Enter your choice: 7
Enter node position to delete: 2
Node Deleted

Enter your choice: 1
Linked List is: 75 50

Enter your choice: 8

** EXIT**

Experiment8:-

Write a program to implement Bubble sort using C.

Program:

```
#include<stdio.h>
void swap(int *,int *);
void bubblesort(int arr[],int size)
{
        int i, j;
        for(i =0;i <size; i++)
      {
        for(j =0; j < size-i;j++)
        {
                if(arr[j]>arr[j+1])
                swap(&arr[j],&arr[j+1]);
        }
      }
}
void swap(int *a,int *b)
{
        int temp;
        temp=*a;
        *a=*b;
        *b =temp;
}
int main()
{
        int array[100],i,size;
        printf("How many numbers you want to sort:");
        scanf("%d", &size);
        printf("\n Enter %d numbers:",size);
        for (i = 0; i < size; i++)
        scanf("%d", &array[i]);
        bubblesort(array, size);
        printf("\nSorted array is");
        for (i = 0; i < size; i++)
        printf("  %d",array[i]);
        return 0;
}
```

OUTPUT:-

```
How many numbers you want to sort: 5
Enter 5 numbers : 50  20  15  30 11
Sorted array is : 11  15  20  30  50
```

Experiment9:-

Write a program to implement Quick sort using C.

Program:

```
#include <stdio.h>
#include<stdlib.h>
int quickSort(int *arr, int low, int high)
{
int i = low, j =high;
int pivot=arr[(low+high)/2];
while (i <= j)
{
        while(arr[i]<pivot)
                i++;
        while(arr[j]>pivot)
                j--;
    if(i <=j)
   {
        int temp=arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        i++;
        j--;
   }
}
if (low < j)
        quickSort(arr,low,j);
if (i < high)
        quickSort(arr,i,high);
return 0;
}
int main()
{
        int i,j;
        puts("Enter the number of elements in the array:");
        int n;
        scanf("%d",&n);
        int arr[n];
        puts("Enter the elements of the array:");
        for (i = 0; i < n; i++)
        {
                printf("arr[%d]:",i);
                scanf("%d",&arr[i]);
        }
        int low =0;
        int high =n -1;
        int pivot =arr[high];
        int k =low -1;
for(j =low; j <high; j++)
{
        if(arr[j] <= pivot)
```

```
            {
            k++;
    int temp=arr[k];
            arr[k] = arr[j];
            arr[j] = temp;
            }
}
int temp = arr[k + 1];
arr[k+1]=arr[high];
arr[high] = temp;
int pi = k + 1;
quickSort(arr, low, pi - 1);
quickSort(arr,pi+1,high);
puts("The sorted array is:");
for (i = 0; i < n; i++)
{
printf(" %d", arr[i]);
}
return 0;
}
```

OUTPUT:-

Enter the number of elements in the array:5

Enter the elements of the array:

arr[0]:45

arr[1]:11

arr[2]:30

arr[3]:65

arr[4]:70

The sorted array is:

11 30 45 65 70

Experiment-10:-

Write a program to implement Binary tree traversal using C.

Program:

```
#include<stdio.h>
#include<stdlib.h>
/*A binary tree node has data, pointer to left child and a pointer to right child */
struct node{ int data;
struct node* left;
struct node* right;
};

/*Helper function that allocates a new node with the given data and NULL left and right pointers.
*/
struct node* newNode(int data)
{
struct node* node=(struct node*)malloc(sizeof(struct node));
node->data=data;
node->left = NULL;
node->right=NULL;
return (node);
}

/*Given a binary tree, print its nodes according to the "bottom-up" post order traversal. */
void printPostorder(struct node* node)
 {
if (node == NULL)
return;
// first recur on left subtree
printPostorder(node->left);

//then recur on right subtree
printPostorder(node->right);

// now deal with the node
printf(" %d",node->data);
}
/*Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node)
 {
if(node==NULL) return;
/*first recur onleft child */
 printInorder(node->left);
printf(" %d",node->data);

printInorder(node->right);
}

void printPreorder(struct node*node)
{
 if (node == NULL)
```

```
return;
printf(" %d ", node->data);
printPreorder(node->left);
printPreorder(node->right);
}
int main() {
struct node* root=newNode(1);
root->left = newNode(2);
root->right = newNode(3);
root->left->left=newNode(4);
root->left->right=newNode(5);

printf("\n Preorder traversal of binary tree is\n");
printPreorder(root);

printf("\n Inorder traversal of binary tree is\n");
printInorder(root);

printf("\nPostorder traversal of binary tree is\n");
printPostorder(root);

getchar();
return 0;
}
```

OUTPUT:-

```
Preorder traversal of binary tree is
1 2 4 5 3
Inorder traversal of binary tree is
4 2 5 1 3
Postorder traversal of binary tree is
4 5 2 3 1
```

Experiment:-11

Write a program to implement Linear Search using C.

Program:

```
#include<stdio.h>
void main()
{
int num;
int i,keynum,found= 0;
printf("Enter the number of elements");
scanf("%d", &num);
int array[num];
printf("Enter the elements one by one\n");
for(i=0;i<num;i++)
{
        scanf("%d",&array[i]);
}
printf("Enter the element to be searched");
scanf("%d", &keynum);
for(i=0;i<num;i++)
{
if(keynum==array[i] )
{
found=1; break;
}
}
if(found==1)
printf("Element is present in the array at position %d",i+1);
 else
printf("Element is not present in the array\n");
}
```

OUTPUT:-

```
Enter the number of elements 6
Enter the elements one by one 4
6
1
2
5
3
Enter the element to be searched 6
Element is present in the array at position 2
```

Experiment12:-

Write a program to implement Binary Search using C.

Program:

```
#include<stdio.h>
int binarySearch(int a[],int beg,int end,int val)
{
	int mid;
	if(end>=beg)
	{
		mid=(beg+end)/2;
	if(a[mid] == val)
	{
	return mid+1;
}
else if(a[mid]<val)
{
	return binarySearch(a,mid+1,end,val);
}
else
{
	return binarySearch(a,beg,mid-1,val);
}
}
	return -1;
}
int main()
{
	int i;
int a[]={11,14,25,30,40,41,52,57,70};//givenarray
int val = 40; // value to be searched
int n=sizeof(a)/ sizeof(a[0]);// sizeof array
int res=binarySearch(a,0,n-1,val);//Store result
printf("The elements of the array are - ");
for(i=0;i<n;i++)
printf("%d ", a[i]);
printf("\nElement to be searched is-%d",val);
 if (res == -1)
printf("\nElement is not present in the array");
	else
printf("\nElement is present at %d position of array",res);
return 0;
}
```

OUTPUT:-

```
The elements of the array are-11   14   25   30   40  41   52   57   70
 Element to be searched is - 40
Element is present at 5 position of array
```