DATA STRUCTURES AND COMPUTER ALGORITHMS LAB

GROUP - A

1. **STACK USING ARRAY**
2. **QUEUE USING ARRAY**
3. **STACK USING LINKED LIST**
4. **QUEUE USING LINKED LIST**
5. **BINARY TREE TRAVERSAL**
6. **INFIX TO POSTFIX CONVERSION**
7. **BINARY SEARCH TREE**

GROUP – B

8. **LINEAR SEARCH**
9. **BINARY SEARCH**
10. **BUBBLE SORT**
11. **INSERTION SORT**
12. **SELECTION SORT**
13. **MERGE SORT**
14. **QUICK SORT**

1. STACK USING AN ARRAY

SOURCE CODE:

```
#include<stdio.h >
#include<conio.h>
int
stack[100],top=0,no;
void push();
void pop();
void display();
 void main()
        {
        int choice;
          clrscr();
           printf("\n\t\t C PROGRAM USING STACK AS AN ARRAY");
             printf("\n\t\t *  ********** ****** ****** *** *** *******");
          printf("\n1.Push");
          printf("\n2.Pop");
          printf("\n3.Display");
          printf("\n4.Quit");
          while(1)
                  {
           printf(" \nEnter your choice: ");
          scanf("%d",&choice);
           switch(choice)
         {
        case 1:
                    push();
                    break;
          case 2:
                    pop();
                    break;
        case 3:
                    display();
                    break;
        case 4:
                    exit(0);
                    break;
          default:
                  printf("\nInvaild choice");
                   }
           }
        }
void push()
                  {
                          if(top==3)
                          {
                                  printf("\n stack is full");
                          }
                          else
                          {
                                 printf("\nEnter the number : ");
                                  scanf("%d",&no);
                                  stack[top]=no;
```

```
                top++;
            }
        }
 void pop()
        {
            int n;
            if(top<=0)
            {
                printf("\n stack is empty");
            }
            else
            {
                 n=stack[top];
                printf("\n %d is poped from the stack\n",n);

            }
        }
void display()
        {
            int i;
            if(top==0)
            {
                printf("\nstack empty");
            }
            else
            {
      printf("\nElement in the stack is ");
      for(i=--top;i>=0;i--)
                printf("\n%d",stack[i]);
            }
        }
```

OUTPUT:

```
                C PROGRAM USING STACK AS AN ARRAY
                ~~ ~~~~~~~~ ~~~~~~ ~~~~~~ ~~ ~~ ~~~~~~

1.Push
2.Pop
3.Display
4.Quit
Enter your choice: 1

Enter the number : 3

Enter your choice: 1

Enter the number : 2

Enter your choice: 1

Enter the number : 1

Enter your choice: 1

 stack is full
Enter your choice: 3_


Element in the stack is
1
2
3
Enter your choice: 2

 1 is poped from the stack

Enter your choice: 3

Element in the stack is
2
3
Enter your choice: 2

 2 is poped from the stack

Enter your choice: 3

Element in the stack is
3
Enter your choice: 2

 stack is empty
Enter your choice: 5
Invalid choice

Enter your  choice: 4
```

2. QUEUE USING ARRAYS

SOURCE CODE:

```
#include<stdio.h>
 #include<conio.h>
 int q[5],front=3,rear=3,no;
 void queue();
void dequeue();
void display();
 void main()
{
int c;
   clrscr();
printf("\n\t\tQUEUE PROGRAM USING ARRAY");
printf("\n\t\t-----------------------------------------------");
printf("\n1.Add element");
printf("\n2.Remove element");
printf("\n3.Display all");
printf("\n4.Quit");
   while(1)
          {
                printf("\nEnter your choice : ");
                 scanf("%d",&c);
                 switch(c)
                 {
                   case 1:
                              queue();
                              break;
                   case 2:
                              dequeue();
                              break;
                   case 3:
                              display();
                              break;
                   case 4:
                              exit(0);
                              break;
                   default:
                                    printf("\n Invaild choice");
                 }
          }
}
 void queue()
{
          if(rear<0)
                 printf("\n Queue is full");
           else
           {
                printf ("\n Enter the number : ");
                scanf("%d",&no);
                   q[rear]=no;
                 rear--;
```

```
            }
}
void dequeue()
{
            int no;
if(front==rear)
printf("\n queue is empty");
            else
            {
                        no=q[front--];
                       printf("\n %d is poped from the queue",no);
            }
}
void display()
{
            int i;
           if(front==rear)
                        printf("\nqueue is empty");
            else
            {
   printf("\n Element in the queue\n");
   for(i=front;i!=rear;i--)
                        printf("\n%d",q[i]);
            }
}
```

OUTPUT:

```
                QUEUE PROGRAM USING ARRAY
                ~~~~~ ~~~~~~~ ~~~~~ ~~~~~

1.Add element
2.Remove element
3.Display all
4.Quit
Enter your choice : 1

 Enter the number : 1

Enter your choice : 1

 Enter the number : 2

Enter your choice : 1

 Enter the number : 3

Enter your choice : 1

 Enter the number : 4

Enter your choice : 1_

 Queue is full
Enter your choice : 3

 Element in the queue

1
2
3
4
Enter your choice : 2

 1 is poped from the queue
Enter your choice : 2

 2 is poped from the queue
Enter your choice : 2

 3 is poped from the queue
Enter your choice : 2

 4 is poped from the queue
Enter your choice : 2

 queue is empty
Enter your choice : 3
```

queue is empty

Enter your choice : 5

Invalid choice

Enter your choice : 4

3. STACK USING LINKED LIST

SOURCE CODE*:*

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void push(); void pop();
void display();
struct stack
{
  int no;
struct stack*next;
}*p,*q,*t,*head=NULL;
void main()
{
int c;
   clrscr();
printf("\n\t\t C PROGRAM USING STACK AS A LINKED LIST");
   printf("\n\t\t**************************************");
   printf("\n1.push()");
   printf("\n2.pop()");
   printf("\n3.display()");
   printf("\n4.quit()");
          while(1)
          {
                  printf("\nEnter your option : ");
                   scanf("%d",&c);
                   switch(c)
                   {
             case 1:
                      push();
                      break;
             case 2:
                      pop();
                      break;
                case 3:
                                display();
                               break;
   default:
             exit(0);
                    }
          }
}
void push()
{
t=malloc(sizeof( struct stack ));
   //print("\n%u",t);
   printf("\nEnter no: ");
   scanf("%d",&t->no);
          if(head==NULL)
          {
                  head=t;
```

```
                t->next=NULL;
        }
        else
        {
  t->next=head;
  head=t;
        }
                printf("\nElement %d is pushed in to the stack",t->no);
}
void pop()
{
        p=head;
        if (head==NULL)
  `     {
                printf("\nStack is empty ");
        }
        else
        {
                head=head->next;
               printf("\nElement %d is popped in to the stack\n",p->no);
                free(p);
        }
}
 void display()
{
        if(head==NULL)
        {
                printf("\nstack is empty");
        }
        else
        {
                p=head;
               printf("\nElement present in the stack : \n");
                while(p!=NULL)
                {
                        printf("%d\n",p->no);
                         p=p->next;
                }
        }
}
```

OUTPUT:

```
                C PROGRAM USING STACK As A LINKED LIST
                * ******* ***** ***** ** * ****** ****
1.push()
2.pop()
3.display()
4.quit()
Enter your option : 1

Enter no: 3

Element 3 is pushed in to the stack
Enter your option : 1

Enter no: 2

Element 2 is pushed in to the stack
Enter your option : 1

Enter no: 1

Element 1 is pushed in to the stack
Enter your option : 3



Element 1 is pushed in to the stack
Enter your option : 3

Element present in the stack :
1
2
3

Enter your option : 2

Element 1 is popped in to the stack

Enter your option : 2

Element 2 is popped in to the stack

Enter your option : 2

Element 3 is popped in to the stack

Enter your option : 2

Stack is empty
Enter your option : 4_
```

4. QUEUE USING LINKED LIST

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
 #include<stdlib.h>
 void enqueue();
void dequeue();
 void display();
 struct queue
{
          int no;
          struct queue*next;
};
struct queue*frontptr,*rearptr,*deleteptr,*p,*t;
void main()
{
int choice;
  frontptr=rearptr=NULL;
  clrscr();
printf("\n\t\t QUEUE AS A LINKED LIST");
  printf("\n\t\t **********************");
  printf("\n1.Add element");
  printf("\n2.Delete element");
  printf("\n3.Display");
printf("\n4.Quit");
  while(1)
          {
  printf("\nEnter your option: ");
  scanf("%d",&choice);
                    switch(choice)
                    {
                              case 1:enqueue();
                                        break;
                              case 2:dequeue();
                                        break;
                              case 3:display();
                      break;
  default:exit(0);
                    }
          }
}
void enqueue()
{
t= malloc (sizeof(struct queue));
  printf("\nEnter no: ");
  scanf("%d",&t->no);    t-
>next=NULL;      if(frontptr==NULL)
          {
                    frontptr=t;
                    rearptr=t;
```

```
            }
            else
            {
   rearptr->next=t;
   rearptr=rearptr->next;
            }
}
void dequeue()
{
            if(frontptr==NULL)
            {
                        printf("\nQueue is empty\n");
                         return;
            }
           deleteptr=frontptr;
            if(deleteptr!=NULL)
            {
                       printf("\nremoved element is %d\n",deleteptr->no);
                         t=frontptr;
                        frontptr=frontptr->next;
                        free(deleteptr);
            }
}
 void  display()
{
            p=frontptr;
             if(p==NULL)
             {
                        printf("\nQueue is empty\n");
                         return;
             }
          printf("\nThe element present in the queue:\n");
             while(p!=NULL)
             {
   printf("\n%d",p->no);
   p=p->next;
            }
}
```

OUTPUT:

```
                QUEUE AS A LINKED LIST
                **********************
1.Add element
2.Delete element
3.Display
4.Quit
Enter your option: 1

Enter no: 1

Enter your option: 1

Enter no: 2

Enter your option: 1

Enter no: 3

Enter your option: 1

Enter no: 4

Enter your option: 3


The element present in the queue:

1
2
3
4
Enter your option: 2

Removed element is 1

Enter your option: 2

Removed element is 2

Enter your option: 2

Removed element is 3

Enter your option: 2

Removed element is 4

Enter your option: 2_
```

Queue is empty

Enter your option: 4

5. BINARY TREE TRAVERSAL

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *left;
struct node *right;
};
void inorder(struct node * root)
{
if(root==NULL)
return;
inorder(root->left);
printf("%d\t",root->data);
inorder(root->right);
}
void preorder(struct node * root)
{
if(root==NULL)
return;
printf("%d\t",root->data);
preorder(root->left);
preorder(root->right);
}
void postorder(struct node * root)
{
if(root==NULL)
return;
postorder(root->left);
postorder(root->right);
printf("%d\t",root->data);
}
Struct node *createnode(value)
{
struct node * newnode=malloc(sizeof(struct node));
newnode->data=value;
newnode->left=NULL;
newnode->right=NULL;
return newnode;
}
struct node *insertleft(struct node * root,int value)
{
root->left=createnode(value);
return root->left;
}
struct node *insertright(struct node * root,int value)
{
root->right=createnode(value);
return root->right;
}
void main()
```

```
{
struct node *root=createnode(1);
insertleft(root,12);
insertright(root,9);
insertleft(root->left,5);
insertright(root->right,6);
clrscr();
printf("\t\t\t BINARY TREE TRAVERSAL \n");
printf("\t\t\t ****** **** ********* \n");
printf("\n INORDER TRAVERSAL \n");
inorder(root);
printf("\n PREORDER TRAVERSAL\n");
preorder(root);
printf(\n POSTORDER TRAVERSAL \n");
postorder(root);
getch();
}
```

OUTPUT:

BINARY TREE TRAVERSAL
****** **** ***************

INORDER TRAVERSAL

5 12 1 9 6

PREORDER TRAVERSAL

1 12 5 9 6

POSTORDER TRAVERSAL

5 12 6 9 1

6. INFIX TO POSTFIX CONVERSION

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
char stack[20];
int top=-1;
void push(char x)
{
Stack[++top]=x;
}
Char pop()
{
if(top==-1)
return -1;
else
return stack[top--];
}
int priority(char x)
{
if(x==’(‘)
return 0;
if(x==’+’||x==’-‘)
return 1;
if(x==’*’||x==’/’)
return 2;
}
void main()
{
char exp[20];
char *e,x;
clrscr();
printf(“\t\t\t INFIX TO POSTFIX CONVERSION\n”);
printf(“\t\t\t **********************\n”);
printf(“Enter the Expression\n”);
scanf(“%s”,exp);
e=exp;
while(*e!=’\0’)
{
if(isalnum(*e))
printf(“%c”,*e);
else if(*e==’(‘)
push(*e)
else if(*e==’)’)
{
while((x=pop()!=’(‘)
printf(“%c”,x)
}
else
{
while(priority(stack[top]>=priority(*e))
printf(“%c”,pop());
push(*e);
}
e++;
}
```

```
while(top!=-1)
{
printf("%c",pop());
}
getch();
}
```

OUTPUT

INFIX TO POSTFIX CONVERSION

Enter the Expression

(a+b)*c

ab+c*

7.BINARY SEARCH TREE

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct  node
{
struct node *left;



int data;
struct node *right;
};
void main()
{
void insert(struct node **,int);
void inorder(struct node*);
void postorder(struct node*);
void preorder(struct node*);
struct node *ptr;
int no,i,num;
ptr=NULL;
ptr->data=NULL;
clrscr();
printf("\t\t\t BINARY TREE TRAVERSAL\n");
printf("Enter the Number of Nodes to be Added\n");
scanf("%d",&no);
for(i=0;i<=no;i++)
{
printf("Enter the node\n");
scanf("%d",&num);
insert(&ptr,num);
}
printf("\nInorder:");
inorder(ptr);
printf("\nPostorder:");
postorder(ptr);
printf("\nPreorder:");
preorder(ptr);
getch();
}
void insert(struct node **p,int num)
{
if(((*p)==NULL)
{
(*p)=malloc(sizeof(struct node));
(*p)->left=NULL;
(*p)->right=NULL;
(*p)->data=num;
return;
}
else
{
if(num==(*p)->data)
```

```
{
return;
}
if(num<(*p)->data)
{
printf("\n Directed to Left Link\n"):
insert(&((*p)->left),num);
}
else
{
printf("\n Directed to Right Link\n");
insert(&((*p)->right),num);
}}
return;
}
void inorder(struct node *p)
{
if(p!=NULL)
{
inorder(p->left);
printf("\t%d",p->data);
inorder(p->right);
}
else
return;
}
void preorder(struct node *p)
{
if(p!=NULL)
{
printf("\t%d",p->data);
preorder(p->left);
preorder(p->right);
}
else
return;
}
void postorder(struct node *p)
{
if(p!=NULL)
{
postorder(p->left);
postorder(p->right);
printf("\t%d",p->data);
}
else
return;
}
```

OUT PUT

BINARY SEARCH TREE
****** ****** **********

Enter the Number of Nodes to be added

3

Enter the Node

45

Enter the Node

78

Directed to Right Link

Enter the Node

23

Directed to Left Link

Inorder: 23 45 78

Postorder: 23 78 45

Preorder: 45 23 78

8.**LINEAR SEARCH**

SOURCE CODE*:*

```
#include<iostream.h>
 #include<conio.h>
 void main()
 {
   clrscr();
    int arr[10],n,i,num,c=0,pos;
   cout<<"\n\t\tLINEAR SEARCH USING AN ARRAY";
   cout<<"\n\t\t--------------------------------------------------";
   cout<<"\nEnter the array size: ";
cin>>n;
   cout<<"\nEnter array
element: ";
for(i=0;i<n;i++)
cin>>arr[i];
   cout<<"\nEnter the number to be search: ";
cin>>num;
   for(i=0;i<n;i++)
 {
          if(arr[i]==num)
   {
 c=1;
  pos=i+1;
           break;
   }  }
 if(c==0)
          cout<<"\nNumber not found...!";
 else
         cout<<"\n "<<num<<" found at position "<<pos;
getch();
}
```

OUTPUT 1:

```
               LINEAR SEARCH USING AN ARRAY
               ^^^^^^ ^^^^^^ ^^^^^ ^^ ^^^^^

Enter the array size: 5

Enter array element: 5 4 3 2 1

Enter the number to be search: 5

 5 found at position 1_
```

OUTPUT 2:

```
               LINEAR SEARCH USING AN ARRAY
               ^^^^^^ ^^^^^^ ^^^^^ ^^ ^^^^^

Enter the array size: 5

Enter array element: 5 4 3 2 1

Enter the number to be search: 6

Number not found...!_
```

9.BINARY SEARCH

SOURCE CODE:

```
#include<iostream.h>
#include<conio.h>
 void main()
{
          clrscr();
int a[10],i,no,sno,j,temp,top,buttom,mid;
   cout<<"\n\t\tBINARY SEARCH ";
cout<<"\n\t\t------------------------ ";
cout<<"\nEnter the number do you want to feed: ";
           cin>>no;
          cout<<"\nEnter the number: ";
           for(i=0;i<no;i++)
           {
                    cin>>a[i];
           }
          cout<<"\nEnter the search number: ";
           cin>>sno;
for(i=0;i<no-1;++i)
for(j=i+1;j<no;++j)
            if(a[i]>a[j])
            {
    temp=a[i];
a[i]=a[j];
a[j]=temp;
            }
         cout<<"\nSorted array in ascending order :\n\n";
  for(i=0;i<no;++i)
     cout<<a[i]<<"\n";
buttom=0;
  top=no-1;
          while(top!=buttom+1)
           {
              mid=(buttom+top)/2;
               if(a[mid]<=sno)
               {
                       buttom=mid;
    }
else
               {
                       top=mid;
               }
           }
          if(a[buttom]==sno)
           {
                  cout<<"\n "<<sno<<" Number is present";
           }
           else
           {
```

```
                cout<<"\n "<<sno<<" Number is not present";
        }
        getch();
    }
```

OUTPUT 1:

```
                BINARY SEARCH
                ------ ------
Enter the number do you want to feed: 5

Enter the number: 5 4 3 2 1

Enter the search number: 4

Sorted array in ascending order :

1
2
3
4
5

 4 Number is present_
```

OUTPUT 2:

```
                BINARY SEARCH
                ------ ------
Enter the number do you want to feed: 5

Enter the number: 5 4 3 2 1

Enter the search number: 8

Sorted array in ascending order :

1
2
3
4
5

 8 Number is not present
```

10.BUBBLE SORT

SOURCE CODE:

```
#include<iostream.h>
#include<conio.h>
 int noofitems,X[100],i,j;
void sort();
void display();
 void input();
 void main()
{
int i;
 clrscr();
cout<<"\n\t\tBUBBLE SORT";
cout<<"\n\t\t**************";
cout<<"\nEnter how many element do you want to feed:";
   cin>>noofitems;
cout<<"\nEnter "<<noofitems<<" elements: ";
   for(i=0;i<noofitems;++i)
            {
                     cin>>X[i];
            }
input();
   sort();
   display();
            getch();
}
void input()
{
cout<<"\nElements are:\n\n";
   for(i=0;i<noofitems;++i)
            cout<<"\t"<<X[i];
}
void display()
{
cout<<"\n\nSorted elements are:\n\n";
   for(i=0;i<noofitems;++i)
   cout<<"\t"<<X[i];
}
void sort()
{
            int swap=1,temp;
            for(i=0;i<noofitems && swap==1;++i)
             {
                      swap=0;
                      for(j=0;j<noofitems-(i+1);++j)
                               if(X[j]>X[j+1])
                               {
                          temp=X[j];
               X[j]=X[j+1];
```

```
        X[j+1]=temp;
        swap=1;
                        }
            }
    }
```

OUTPUT:

```
                BUBBLE SORT
                ***********
Enter how many element do you want to feed:5

Enter 5 elements: -8 4 1 7 -2

Elements are:

        -8      4       1       7       -2

Sorted elements are:

        -8      -2      1       4       7
```

11.INSERTION SORT

SOURCE CODE:

```
#include<iostream.h>
 #include<conio.h>
 int noofitems,x[100],i;
 void sort();
 void display();
 void main()
{
clrscr();  cout<<"\n\t\tINSERTION SORT";
  cout<<"\n\t\t****************";
cout<<"\nEnter how many element do you want to feed: ";
  cin>>noofitems;
cout<<"\nEnter "<<noofitems<<" elements:";
  for(i=0;i<noofitems;++i)
          {
                  cin>>x[i];
          }
sort();
  display();
  getch();
}
void display()
{
         cout<<"\nSorted elements are:";
          for(i=0;i<noofitems;++i)
                  cout<<"\n"<<x[i];
}
void sort()
{
          int s[100],j,t;
          s[0]=x[0];
          for(i=0;i<noofitems;++i)
          {
                  t=x[i];
                  j=i-1;
                  while((s[j]>t) && (j>=0))
                  {
                          s[j+1]=s[j];
                          j--;
                  }
                  s[j+1]=t;
          }
          for(i=0;i<noofitems;++i)
          {
                  x[i]=s[i];
          }
}
```

OUTPUT:

```
                INSERTION SORT
                ***************
Enter how many element do you want to feed: 4

Enter 4 elements:22 44 11 33

Sorted elements are:
11
22
33
44_
```

12.SELECTION SORT

SOURCE CODE:

```
#include<iostream.h>
 #include<conio.h>
 class selection
{
public:
double x[100],n,i;
   void get();
void sort();
 void display();
};
void main()
{
int i;
clrscr();  cout<<"\n\t\tSELECTION SORT";
   cout<<"\n\t\t~~~~~~~~~~~~ ~~~~";
selection a;
   a.get();
            a.sort();
            a.display();
            getch();
}
void selection::sort()
{
            int t,j,k,low;
```

```
        for(j=0;j<n-1;++j)
        {
                low=j;
                for(k=j+1;k<n;++k)
                {
                        if(x[k]<=x[low])
                        {
                                low=k;
                        }
                        t=x[j];
          x[j]=x[low];
  x[low]=t;
                }
        }  }
void selection::display()
{
        int i;
cout<<"\nSorted elements are:\n ";
  for(i=0;i<n;++i)
        {
                cout<<"\n "<<x[i];
        }
}
void selection::get()
{
       cout<<"\nEnter how many element you want to feed: ";
         cin>>n;
cout<<"\nEnter "<<n<<" element: ";
  for(i=0;i<n;++i)
        {
                cin>>x[i];
        }   }
```

OUTPUT:

```
                SELECTION SORT
                ~~~~~~~~~ ~~~~

Enter how many element you want to feed: 7

Enter 7 element: 0 -1 4 2 7 3 -5

Sorted elements are:

 -5
 -1
 0
 2
 3
 4
 7_
```

13.MERGE SORT

SOURCE CODE:

```
#include<iostream.h>
#include<conio.h>
int a[100];
 void merge(int,int,int);
 void ms(int low,int high)
{
int mid;
  if(low<high)
        {
  mid=(low+high)/2;
  ms(low,mid);
  ms(mid+1,high);
  merge(low,mid,high);
        }
}
void merge(int low,int mid,int high)
{
        int h,i,j,b[100],k;
h=low;
  i=low;
        j=mid+1;
```

```
        while((h<=mid) && (j<=high))
        {
                if(a[h]<=a[j])
                {
                        b[i]=a[h];
          h++;
  }               else
                {
                        b[i]=a[j];
          j++;
  }
                i++;
        }
        if(h>mid)
        {
                for(k=j;k<=high;k++)
                {
                        b[i]=a[k];
                        i++;
                }
        }
        else
        {
                for(k=h;k<=mid;k++)
                {
                        b[i]=a[k];
                        i++;
                }
        }
                for(k=low;k<=high;k++)
                        a[k]=b[k];
} void main()
{
int num,i;
clrscr();  cout<<"\n\t\tMERGE SORTING";
  cout<<"\n\t\t**********”;
cout<<"\nEnter no.of.element do you want to sort: ";
        cin>>num;
       cout<<"\nEnter " <<num<<" element: "<<"\n";
        for(i=1;i<=num;i++)
        {
                cin>>a[i];
        }
        ms(1,num);
       cout<<"\nAfter merge sort,sorted array is:\n";
        for(i=1;i<=num;i++)
        {
                cout<<a[i]<<" ";
        }
cout<<"\n";
  getch();
}
```

OUTPUT:

```
                MERGE SORTING
                ~~~~~ ~~~~~~~
Enter no.of.element do you want to sort: 3

Enter 3 element:
66
22
44

After merge sort,sorted array is:
22 44 66
```

14.QUICK SORT

SOURCE CODE:

```
#include<iostream.h>
 #include<conio.h>
void quick(int p,int q);
 int par(int m,int j);
void intercha(int i,int j);
 int a[100];
 void main()
{
int p,q,n,in=0;
clrscr();
cout<<"\n\t\tQUICK SORTING";
  cout<<"\n\t\t_____";
        cout<<"\nEnter the limit: ";
         cin>>n;
        cout<<"\nEnter the element: ";
         for(int i=1;i<=n;i++)
         {
                  cin>>a[i];
                  in=in+a[i];
         }
p=1;
  q=n;
a[n+1]=in;
```

```
quick(p,q);
   cout<<"\nSorted
element\n\n";
   for(i=1;i<=n;i++)
            {
                     cout<<a[i]<<"\t";
            }
            getch();
} void quick(int p,int q)
{
            if(p<q)
            {
   int j=par(p,q+1);
   quick(p,j-1);
                     quick(j+1,q);
            }
} int par(int m,int p)
{
int fe,i;
   fe=a[m];
            int j=p;
i=m;
   do       {

   do
                     {
                              i=i+1;
                     }while(a[i]<fe);
                     do
                     {
                              j=j-1;
   }while(a[j]>fe);
   if(i<j)
                     {
                              intercha(i,j);
                     }
}while(i<j);
   a[m]=a[j];
   a[j]=fe;
            return(j);

} void
intercha(int
i,int j)
{
int t=a[i];
   a[i]=a[j];
            a[j]=t;
}
```

OUTPUT:

```
                QUICK SORTING
                @@@@@ @@@@@@@
Enter the limit: 5

Enter the element: 5 -9 3 1 -7

Sorted element

-9      -7      1       3       5
```