**Name:** Purvi Singhvi

**Enrollment No.:** 180099

**PROGRAM-7**

**Aim:** Write an algorithm and program to sort n numbers using Quick sort technique. **Algorithm:**

Partition:

1. x=a[r]
2. i=p-1
3. for(j=p to(r-1))
4. { if(A[j]<=x)
5. {i=i+1
6. Exchange A[i] with A[j]}}
7. Exchange A[i+1] with A[r]
8. Return i+1

Quick\_Sort(A,p,r)

1. if(p<r)
2. { q= Partition(A,p,r)
3. Quick\_Sort(A,p,q-1)
4. Quick\_Sort(A,q+1,r)
5. }

i) Using arrays

**Source Code:**

#include<stdio.h>

#include<conio.h>

int partition(int a[],int p,int r)

{

int l,i,j,t;

i=p-1;

for(j=p;j<r;j++)

{

if(a[j]<=a[r])

{

i++;

t=a[j];

a[j]=a[i];

a[i]=t;

}

}

l=a[i+1];

a[i+1]=a[r];

a[r]=l;

return (i+1);

}

void quick\_sort(int a[],int p,int r)

{

int q;

if(p<r)

{

q=partition(a,p,r);

quick\_sort(a,p,q-1);

quick\_sort(a,q+1,r);

}

}

void main()

{

clrscr();

int a[15],n,i;

printf("Enter the number of elements: ");

scanf("%d",&n);

printf("Enter the elements of array:\n");

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

{

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

}

quick\_sort(a,0,n-1);

printf("\nArray after sorting\n");

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

{

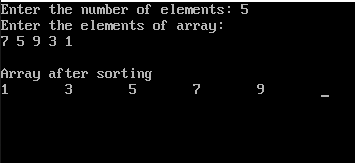
printf("%d\t",a[i]);

}

getch();

}

**Output:**



ii) Using Linked List

**Source Code:**

#include<stdlib.h>

#include<conio.h>

#include<stdio.h>

struct node{

int data;

struct node \*link;

};

int w=0;

struct node \*head,\*n;

void insertion()

{

struct node \*ptr;

ptr=head;

n=(struct node \*)malloc(sizeof(struct node));

n->link=NULL;

printf("Enter the data: ");

scanf("%d",&n->data);

if(head==NULL)

{

head=n;

}

else

{

while(ptr->link!=NULL)

{

ptr=ptr->link;

}

ptr->link=n;

}

w++;

}

void traversal()

{

struct node \*ptr;

ptr=head;

if(head==NULL)

{

printf("List is empty");

}

else

{

while(ptr!=NULL)

{

printf("%d->",ptr->data);

ptr=ptr->link;

}

}

getch();

}

int partition(int p,int r)

{

struct node \*ptr,\*ptr1,\*ptr2;

int m,t,te,temp,a=1,tem=1,e=1;

ptr2=NULL;

ptr1=head;

ptr=head;

m=p;

temp=r;

te=1;

while(te<m)

{

ptr1=ptr1->link;

te++;

}

while(temp>1)

{

ptr=ptr->link;

temp--;

}

while(ptr1!=ptr)

{

if(ptr1->data<=ptr->data)

{

if(ptr2==NULL&&p==1)

{

ptr2=head;

}

else if(ptr2==NULL&&p!=1)

{

ptr2=head;

while(tem<p)

{

ptr2=ptr2->link;

tem++;

}

}

else

{

ptr2=ptr2->link;

}

t=ptr1->data;

ptr1->data=ptr2->data;

ptr2->data=t;

}

ptr1=ptr1->link;

}

if(ptr2==NULL){

ptr2=head;

while(e<p){

ptr2=ptr2->link;

e++;

}

}

else{

ptr2=ptr2->link;

}

t=ptr->data;

ptr->data=ptr2->data;

ptr2->data=t;

ptr=head;

while(ptr!=ptr2){

a++;

ptr=ptr->link;

}

return a;

getch();

}

void quck\_srt(int p,int r)

{

int q;

if(head==NULL){

printf("List is empty");

getch();

}

else{

if(p<r){

q=partition(p,r);

quck\_srt(p,q-1);

quck\_srt(q+1,r);

}} }

void main(){

int c;

L:system("cls");

printf("1. Insertion\n");

printf("2. Traversal\n");

printf("3. Sorting\n");

printf("4.Exit\n");

printf("Enter your choice: ");

scanf("%d",&c);

switch(c){

case 1:

insertion();

goto L;

case 2:

traversal();

goto L;

case 3:

quck\_srt(1,w);

goto L;

case 4:

exit(0);

default:

printf("Invalid choice...Enter your choice again");

getch();

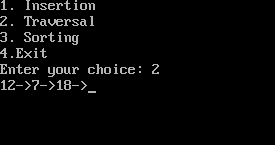
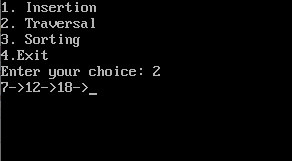
goto L;

}

}

**Output:**

**Before Sorting: After Sorting:**

iii) Using Linked List and without recursion

**Source Code:**

#include<stdlib.h>

#include<conio.h>

#include<stdio.h>

struct node{

int data;

struct node \*link;

};

int w=0;

struct node \*head,\*n;

void insertion(){

struct node \*ptr;

ptr=head;

n=(struct node \*)malloc(sizeof(struct node));

n->link=NULL;

printf("Enter the data: ");

scanf("%d",&n->data);

if(head==NULL){

head=n;

}

else{

while(ptr->link!=NULL){

ptr=ptr->link;

}

ptr->link=n;

}

w++;

}

void traversal(){

struct node \*ptr;

ptr=head;

if(head==NULL){

printf("List is empty");

}

else{

while(ptr!=NULL){

printf("%d->",ptr->data);

ptr=ptr->link;

}

}

getch();

}

int partition(int p,int r){

struct node \*ptr,\*ptr1,\*ptr2;

int m,t,te,temp,a=1,tem=1,e=1;

if(p<r){

ptr2=NULL;

ptr1=head;

ptr=head;

m=p;

temp=r;

te=1;

while(te<m){

ptr1=ptr1->link;

te++;

}

while(temp>1){

ptr=ptr->link;

temp--;

}

while(ptr1!=ptr){

if(ptr1->data<=ptr->data){

if(ptr2==NULL&&p==1){

ptr2=head;

}else if(ptr2==NULL&&p!=1){

ptr2=head;

while(tem<p){

ptr2=ptr2->link;

tem++;

}

}

else{

ptr2=ptr2->link;

}

t=ptr1->data;

ptr1->data=ptr2->data;

ptr2->data=t;

}

ptr1=ptr1->link;

}

if(ptr2==NULL){

ptr2=head;

while(e<p){

ptr2=ptr2->link;

e++;

}

}

else{

ptr2=ptr2->link;

}

t=ptr->data;

ptr->data=ptr2->data;

ptr2->data=t;

ptr=head;

while(ptr!=ptr2){

a++;

ptr=ptr->link;

}

return a;

}

}

void quck\_srt(int p,int r){

int q,stack[30];

int top = -1;

stack[++top]=p;

stack[++top]=r;

while (top >= 0){

r=stack[top--];

p=stack[top--];

q=partition(p,r);

if(p<q-1){

stack[++top]=p;

stack[++top]=q-1;

}

if(q+1<r){

stack[++top]=q+1;

stack[++top]=r;

}

}

}

void main(){

int c;

L:system("cls");

printf("1. Insertion\n");

printf("2. Traversal\n");

printf("3. Sorting\n");

printf("4.Exit\n");

printf("Enter your choice: ");

scanf("%d",&c);

switch(c){

case 1:

insertion();

goto L;

case 2:

traversal();

goto L;

case 3:

quck\_srt(1,w);

goto L;

case 4:

exit(0);

default:

printf("Invalid choice...Enter your choice again");

getch();

goto L;

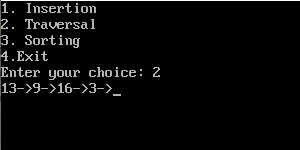
}

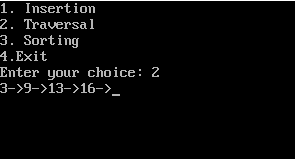
getch();

}

**Output:**

**Before Sorting After Sorting**





**Complexity:**

Best Case: O(n logn)

Worst Case: O(n2)

Average Case: O(n logn)