**7.** **Write an algorithm and program to sort n numbers using Quick sort.**

**i) Using arrays**

**ii) Using linked list**

**iii) Using linked list and without recursion**

**(i)Using arrays:-**

#include<stdio.h>

void quicksort(int a[],int l,int h);

int qpart(int a[],int l,int h);

void main()

{

int n,i,l,h,a[20];

printf("\n enter size=");

scanf("%d",&n);

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

{

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

}

l=0;

h=n-1;

quicksort(a,l,h);

printf("\n sorted array=\n");

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

{

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

}

getch();

}

void quicksort(int a[],int l,int h)

{

int k;

if(l<h)

{

k=qpart(a,l,h);

quicksort(a,l,k);

quicksort(a,k+1,h);

}}

int qpart(int a[],int l,int h)

{

int pivot,i,j,temp;

i=l;

j=h;

pivot=a[l];

while(i<j)

{

while(a[i]<=pivot)

{

i++;

}

while(a[j]>pivot)

{

j--;

}

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

temp=a[l];

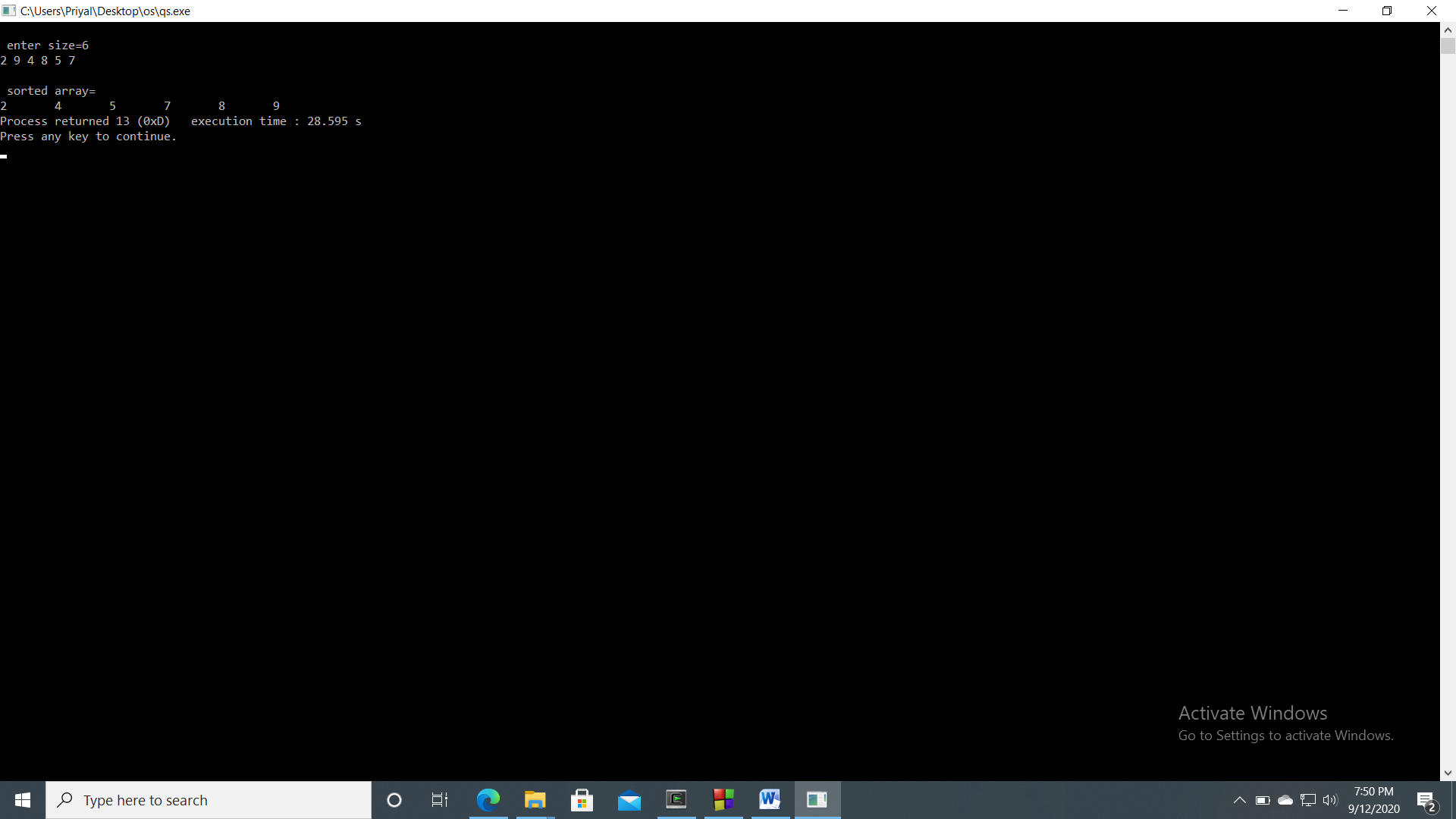
a[l]=a[j];

a[j]=temp;

return(j);

}

**Output:-**



**ii) Using linked list**

#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;

}

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");

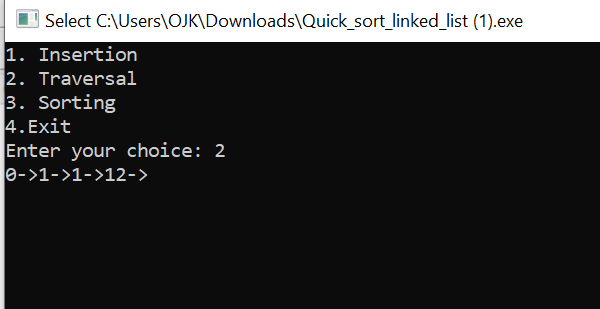
getch();

goto L;

}

}

Output



**iii) Using linked list and without recursion**

#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[r-p+1];

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;

}

}

**Output**

