**PROGRAM-7**

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

ALGORITHM-

quickSort(arr[], low, high)

{

if (low < high)

{

/\* pi is partitioning index, arr[pi] is now

at right place \*/

pi = partition(arr, low, high);

quickSort(arr, low, pi - 1); // Before pi

quickSort(arr, pi + 1, high); // After pi

}

}

1. **Using arrays**

**Source Code-**

#include<stdio.h>

void quicksort(int x[20],int first,int last)

{

int pivot,j,temp,i;

if(first<last)

{

pivot=first;

i=first;

j=last;

while(i<j)

{

while(x[i]<=x[pivot]&&i<last)

i++;

while(x[j]>x[pivot])

j--;

if(i<j)

{

temp=x[i];

x[i]=x[j];

x[j]=temp;

}

}

temp=x[pivot];

x[pivot]=x[j];

x[j]=temp;

quicksort(x,first,j-1);

quicksort(x,j+1,last);

}

}

int main()

{

int x[20],size,i;

printf("\tQuick sort\n");

printf("-----------------------------------\n");

printf(" How many numbers you want to sort?: ");

scanf("%d",&size);

printf("\n Enter %d elements: \n",size);

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

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

quicksort(x,0,size-1);

printf("\n Sorted elements after applying quick sort: \n\n");

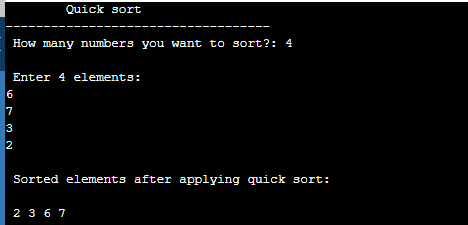
for(i=0;i<size;i++)

printf(" %d",x[i]);

return 0;

}

**OUTPUT-**

****

1. **ii) Using linked list**

**Source Code-**

#include <iostream>

#include <cstdio>

using namespace std;

struct Node

{

int data;

struct Node \*next;

};

void insert(struct Node\*\* head\_ref, int new\_data)

{

struct Node\* new\_node = new Node;

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void print(struct Node \*node)

{

while (node != NULL)

{

cout<<node->data<<" ";

node = node->next;

}

cout<<"\n";

}

struct Node \*getTail(struct Node \*cur)

{

while (cur != NULL && cur->next != NULL)

cur = cur->next;

return cur;

}

struct Node \*partition(struct Node \*head, struct Node \*end,

struct Node \*\*newHead, struct Node \*\*newEnd)

{

struct Node \*pivot = end;

struct Node \*prev = NULL, \*cur = head, \*tail = pivot;

while (cur != pivot)

{

if (cur->data < pivot->data)

{

if ((\*newHead) == NULL)

(\*newHead) = cur;

prev = cur;

cur = cur->next;

}

else

{

if (prev)

prev->next = cur->next;

struct Node \*tmp = cur->next;

cur->next = NULL;

tail->next = cur;

tail = cur;

cur = tmp;

}

}

if ((\*newHead) == NULL)

(\*newHead) = pivot;

(\*newEnd) = tail;

return pivot;

}

struct Node \*quickSortRecur(struct Node \*head, struct Node \*end)

{

if (!head || head == end)

return head;

Node \*newHead = NULL, \*newEnd = NULL;

struct Node \*pivot = partition(head, end, &newHead, &newEnd);

if (newHead != pivot)

{

struct Node \*tmp = newHead;

while (tmp->next != pivot)

tmp = tmp->next;

tmp->next = NULL;

newHead = quickSortRecur(newHead, tmp);

tmp = getTail(newHead);

tmp->next = pivot;

}

pivot->next = quickSortRecur(pivot->next, newEnd);

return newHead;

}

void quickSort(struct Node \*\*headRef)

{

(\*headRef) = quickSortRecur(\*headRef, getTail(\*headRef));

return;

}

int main()

{

struct Node \*start = NULL;

int t,num;

cout<<"Enter the number of elements: ";

cin>>t;

while(t--){

cin>>num;

insert(&start, num);

}

quickSort(&start);

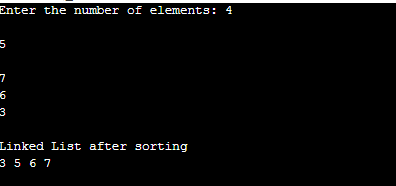
cout << "Linked List after sorting \n";

print(start);

return 0;

}

**OUTPUT-**



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

**SOURCE CODE-**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

int x=0;

struct node \*head, \*n;

void insert(){

struct node \*ptr;

ptr=head;

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

n->next=NULL;

printf("\n Enter the element you want to insert:");

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

if(head==NULL)

{

head=n;

}

else{

while(ptr->next!=NULL){

ptr=ptr->next;

}

ptr->next=n;

}

x++;

}

void traverse(){

struct node \*ptr;

ptr=head;

if(head==NULL){

printf("\n The list is empty");

}

else{

while(ptr!=NULL){

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

ptr=ptr->next;

}

}

getch();

}

int partition(int p, int r){

struct node \*ptr, \*p1, \*p2;

int m,t, y, temp, a=1, u=1, b=1;

if(p<r){

p2=NULL;

p1=head;

ptr=head;

m=p;

temp=r;

y=1;

while(y<m){

p1=p1->next;

y++;

}

while(temp>1)

{

ptr=ptr->next;

temp--;

}

while(p1!=ptr){

if(p1->data<=ptr->data){

if(p2==NULL && p==1){

p2=head;

}

else if(p2==NULL && p!=1){

p2=head;

while(u<p){

p2=p2->next;

u++;

}

}

else{

p2=p2->next;

}

t=p1->data;

p1->data=p2->data;

p2->data=t;

}

p1=p1->next;

}

if(p2==NULL){

p2=head;

while(b<p){

p2=p2->next;

b++;

}

}

else{

p2=p2->next;

}

t=ptr->data;

ptr->data=p2->data;

p2->data=t;

ptr=head;

while(ptr!=p2){

a++;

ptr=ptr->next;

}

return a;

}

}

void quick\_sort(int p,int r)

{

int q,arr[20];

int beg= -1;

arr[++beg]=p;

arr[++beg]=r;

while(beg>=0){

r=arr[beg--];

p=arr[beg--];

q=partition(p,r);

if(p<q-1){

arr[++beg]=p;

arr[++beg]=q-1;

}

if(q+1<r){

arr[++beg]=q+1;

arr[++beg]=r;

}

}

}

void main(){

int ch;

L:system("cls");

printf("\n 1. Insertion");

printf("\n 2. Traversal");

printf("\n 3. Sorting");

printf("\n 4. Exit");

printf("\n Enter the operation you want to perform:");

scanf("%d", &ch);

switch(ch){

case 1: insert();

goto L;

case 2: traverse();

goto L;

case 3: quick\_sort(1,x);

goto L;

case 4: exit(0);

default:

printf("\n Invalid Choice");

getch();

goto L;

}

}

OUTPUT-

