**Heap Sort:**

#include <iostream>

using namespace std;

void max\_heapify(int \*a, int i, int n)

{

    int j, temp;

    temp = a[i];

    j = 2\*i;

    while (j <= n)

    {

        if (j < n && a[j+1] > a[j])

            j = j+1;

        if (temp > a[j])

            break;

        else if (temp <= a[j])

        {

            a[j/2] = a[j];

            j = 2\*j;

        }

    }

    a[j/2] = temp;

    return;

}

void heapsort(int \*a, int n)

{

    int i, temp;

    for (i = n; i >= 2; i--)

    {

        temp = a[i];

        a[i] = a[1];

        a[1] = temp;

        max\_heapify(a, 1, i - 1);

    }

}

void build\_maxheap(int \*a, int n)

{

    int i;

    for(i = n/2; i >= 1; i--)

    {

        max\_heapify(a, i, n);

    }

int main()

{

    int n, i, x;

    cout<<"Enter no of elements of array: "<<endl;

    cin>>n;

    int a[20];

    for (i = 1; i <= n; i++)

    {

        cout<<"Enter element "<<(i)<<endl;

        cin>>a[i];

    }

    build\_maxheap(a,n);

    heapsort(a, n);

    cout<<"\nSorted Array\n";

    for (i = 1; i <= n; i++)

    {

        cout<<a[i]<<endl;

    }

    return 0;

}

**Merge Sort:**

#include <iostream>

using namespace std;

void Merge(int \*a, int low, int high, int mid)

{

int i, j, k, temp[high-low+1];

i = low;

k = 0;

j = mid + 1;

while (i <= mid && j <= high)

{

if (a[i] < a[j])

{

temp[k] = a[i];

k++;

i++;

}

else

{

temp[k] = a[j];

k++;

j++;

}

}

while (i <= mid)

{

temp[k] = a[i];

k++;

i++;

}

while (j <= high)

{

temp[k] = a[j];

k++;

j++;

}

for (i = low; i <= high; i++)

{

a[i] = temp[i-low];

}

}

void MergeSort(int \*a, int low, int high)

int mid;

if (low < high)

{

mid=(low+high)/2;

MergeSort(a, low, mid);

MergeSort(a, mid+1, high);

Merge(a, low, high, mid);

}

}

int main()

{

int n, i;

cout<<"**\n**Enter the number of data element to be sorted: ";

cin>>n;

int arr[n];

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

{

cout<<"Enter element "<<i+1<<": ";

cin>>arr[i];

}

MergeSort(arr, 0, n-1);

cout<<"**\n**Sorted Data ";

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

cout<<"->"<<arr[i];

return 0;

}

**Quick Sort:**

#include <iostream>

using namespace std;

void quick\_sort(int[],int,int);

int partition(int[],int,int);

int main()

{

int a[50],n,i;

cout<<"How many elements?";

cin>>n;

cout<<"\nEnter array elements:";

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

cin>>a[i];

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

cout<<"\nArray after sorting:";

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

cout<<a[i]<<" ";

return 0;

}

void quick\_sort(int a[],int l,int u)

{

int j;

if(l<u)

{

j=partition(a,l,u);

quick\_sort(a,l,j-1);

quick\_sort(a,j+1,u);

}

}

int partition(int a[],int l,int u)

{

int v,i,j,temp;

v=a[l];

i=l;

j=u+1;

do

{

do

i++;

while(a[i]<v&&i<=u);

do

j--;

while(v<a[j]);

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}while(i<j);

a[l]=a[j];

a[j]=v;

return(j);

}

**Selection Sort:**

#include <iostream>

using namespace std;

// Return minimum index

int minIndex(int a[], int i, int j)

{

if (i == j)

return i;

int k = minIndex(a, i + 1, j);

return (a[i] < a[k])? i : k;

}

void recurSelectionSort(int a[], int n, int index = 0)

{

if (index == n)

return;

int k = minIndex(a, index, n-1);

if (k != index)

swap(a[k], a[index]);

recurSelectionSort(a, n, index + 1);

}

int main()

{

int arr[] = {3, 1, 5, 2, 7, 0};

int n = sizeof(arr)/sizeof(arr[0]);

recurSelectionSort(arr, n);

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

cout << arr[i] << " ";

cout << endl;

return 0;

}