Task 1

#include<iostream>

#include<algorithm>

using namespace std;

// partition function

int partition(int arr[],int first, int last){

int mid =(first+last)/2;

swap(arr[mid],arr[last]);

int pivot=arr[last];

int i = first-1;

int j = first;

for(j=first;j<last;j++){

if(arr[j]<pivot){

i++;

swap(arr[i],arr[j]);

}

}

swap(arr[i+1],arr[last]);

return i+1;

}

//Quick sort function

void quicksort(int arr[],int first, int last){

if(first>=last){

return;

}

// partition function

int pi = partition(arr,first,last);

quicksort(arr,first,pi-1);

quicksort(arr,pi+1,last);

}

int main(){

int arr[]={5,6,2,-2,6,10,12,15};

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

// cout<<n;

quicksort(arr,0,n-1);

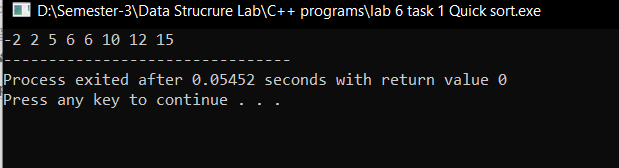
for(int i=0;i<n;i++){

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

}

return 0;

}



Task 2  
Ascending order

#include<iostream>

using namespace std;

int getmax(int arr[],int n){

int max=arr[0];

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

if(arr[i]>max){

max = arr[i];

}

}

return max;

}

void countsort(int arr[],int n,int exp){

int output[n];

int i,count[10]={0};

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

count[(arr[i]/exp)%10]++;

}

for(i=1;i<10;i++){

count[i]+=count[i-1];

}

for(i=n-1;i>=0;i--){

output[count[(arr[i]/exp)%10]-1]=arr[i];

count[(arr[i]/exp)%10]--;

}

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

arr[i]=output[i];

}

}

// Radix sort

void radixsort(int arr[], int n){

int m = getmax(arr,n);

for(int exp=1;m/exp>0;exp\*=10){

countsort(arr,n,exp);

}

}

int main(){

int arr[]={170,45,75,90,802,24,2,66};

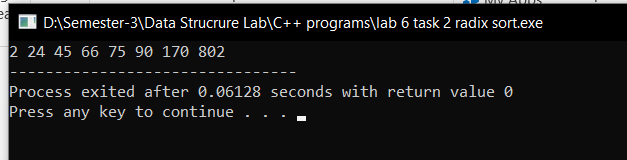
int n= sizeof(arr)/sizeof(arr[0]);

radixsort(arr,n);

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

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

}

}

Descending Order

#include<iostream>

using namespace std;

int getmax(int arr[],int n){

int max = arr[0];

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

if(arr[i]>max){

max = arr[i];

}

}

return max;

}

void countsort(int arr[],int n, int pos){

int output[n];

int count[10]={0};

//counting the occurance

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

int digit =(arr[i]/pos)%10;

count[digit]++;

}

// comulative freq in rev order

for(int i=8;i>=0;i--){

count[i]+=count[i+1];

}

// output array

for(int i=n-1;i>=0;i--){

int digit = (arr[i]/pos)%10;

output[count[digit]-1]=arr[i];

count[digit]--;

}

//copying to the orginal array

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

arr[i]=output[i];

}

}

void radixsort(int arr[],int n){

int max = getmax(arr,n);

for(int pos=1;(max/pos)>0;pos\*=10){

countsort(arr,n,pos);

}

}

int main(){

int arr[]={170,45,75,90,802,24,2,66};

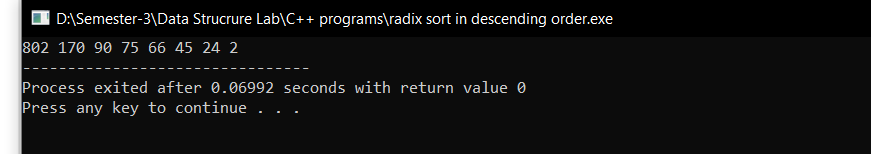
int n=sizeof(arr)/sizeof(arr[0]);

radixsort(arr,n);

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

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

}

}

Task 3

#include<iostream>

using namespace std;

//Merge function

void merge(int arr[],int left,int mid,int right){

int an = mid-left+1;

int bn = right-mid;

// creating two temp arrays

int a[an];

int b[bn];

// storing the values

for(int i=0;i<an;i++){

a[i]=arr[left+i];

}

for(int j=0;j<bn;j++){

b[j]=arr[mid+1+j];

}

int i=0;

int j=0;

int k=left;

while(i<an&&j<bn){

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

arr[k++]=a[i++];

}else{

arr[k++]=b[j++];

}

}

while(i<an){

arr[k++]=a[i++];

}

while(j<bn){

arr[k++]=b[j++];

}

}

// Merge sort

void mergesort(int arr[],int left,int right){

if(left>= right){

return;

}

int mid =(left+right)/2;

mergesort(arr,left,mid);

mergesort(arr,mid+1,right);

merge(arr,left,mid,right);

}

int main(){

int arr[]={3,1,6,8,4,5,7,2};

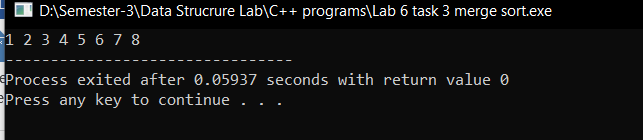
int n = sizeof(arr)/sizeof(arr[0]);

mergesort(arr,0,n-1);

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

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

}

}

Task 4

#include<iostream>

using namespace std;

void binarysearch(int arr[],int n,int target){

int lo =0;

int hi= n-1;

while(lo<=hi){

// calculating the mid point

int mid =(lo+hi)/2;

if(arr[mid]==target){

cout<<"Position is "<<mid;

return ;

}else if(arr[mid]<target){

// discarding the left of mid

lo = mid+1;

}else if(arr[mid]>target){

// discarding the right of mid

hi = mid-1;

}

}

cout<<"Not in the list";

}

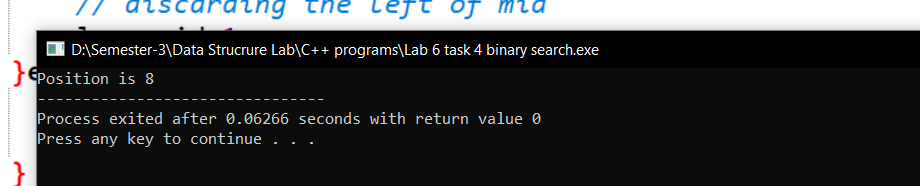
int main(){

int arr[]={1,3,12,14,23,34,55,65,79,75,78};

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

binarysearch(arr,n,79);

}



Task 5

#include<iostream>

using namespace std;

void binarysearch(int arr[],int n,int target){

int lo =0;

int hi= n-1;

while(lo<=hi){

// calculating position

int mid=lo+((hi-lo)/(arr[hi]-arr[lo]))\*(target-arr[lo]);

if(arr[mid]==target){

cout<<"Position is "<<mid;

return ;

}else if(arr[mid]<target){

// discarding the left of mid

lo = mid+1;

}else if(arr[mid]>target){

// discarding the right of mid

hi = mid-1;

}

}

cout<<"Not in the list";

}

int main(){

int arr[]={1,3,7,10,14,15,16,18,20,21,22,23,25,33,35,42,45,47,50,52};

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

binarysearch(arr,n,33);

}