19/08/2012

Experiment No.:5a

MERGE SORT

# AIM:

# Implementation of Merge Sort on an array

# ALGORITHM:

# 

The following is the algorithm for merge sort on an array A of with pper bound ub and lower bound lb.

merge\_sort (A, lb, ub)

{

if(lb<ub)

{

mid=(ub+lb)/2

merge\_sort(A, lb, mid)

merge\_sort(A, mid+1, ub)

merge(A, lb, mid, ub) /\* defined following \*/

}

}

merge((A, lb, ub)

{

i=lb

j=mid+1

while(i≤mid AND j≤ub)

{

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

{

temp[k]=A[i++] /\* temp is an auxiliary array \*/

i=i+1

}

else

{

temp[k]=A[j]

j=j+1

}

k=k+1

}

while(i≤mid)

{

temp[k]=A[i]

i=i+1

k=k+1

}

while(j≤ub)

{

temp[k]=A[j]

j=j+1

k=k+1

}

for i = lb to ub

A[i]=temp[i-lb];

}

# SOURCE CODE:

/\* Author : Rakesh Malik

\* Date : 11.08.2012

\* Merge Sort

\*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void merge(int list[],int lb,int mid,int ub) {

int temp[ub-lb+1],i,j,k=0,m;

for(i=lb,j=mid+1;i<=mid&&j<=ub;)

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

temp[k++]=list[i++];

else

temp[k++]=list[j++];

for(;i<=mid;i++)

temp[k++]=list[i];

for(;j<=ub;j++)

temp[k++]=list[j];

for(i=lb;i<=ub;i++)

list[i]=temp[i-lb];

}

int merge\_sort(int list[], int lb, int ub) {

if(lb<ub) {

int mid=(ub+lb)/2;

merge\_sort(list, lb, mid);

merge\_sort(list, mid+1, ub);

merge(list, lb, mid, ub);

}

}

int main() {

int n, i;

printf("MERGE SORT\n\n");

printf("Enter number of elements : ");

scanf("%d", &n);

printf("Enter elements (space seperated) : ");

int list[n];

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

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

merge\_sort(list, 0, n-1);

printf("\nAfter sorting : ");

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

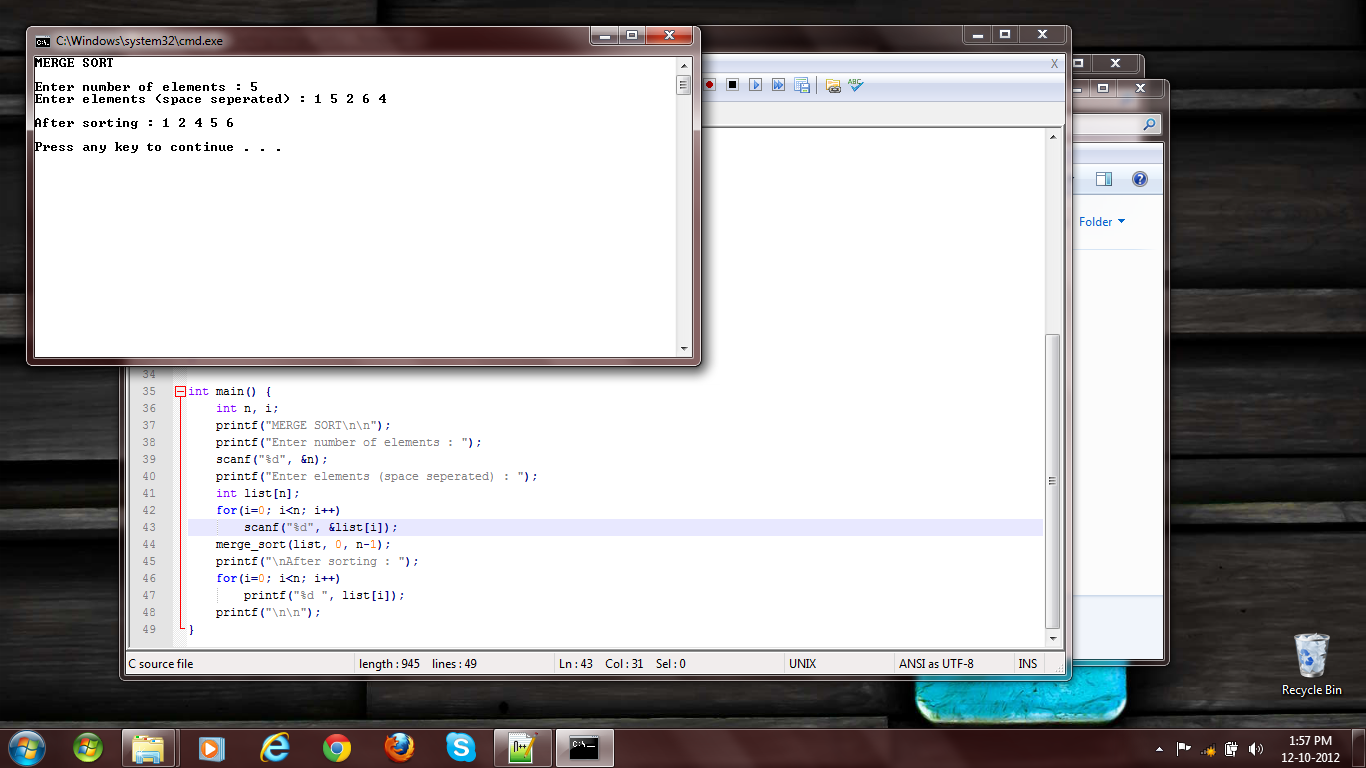


Fig 1: Merge Sort

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

printf("\n\n");

}

# RESULT:

The program has run successfully and given result as expected.