Practical 3

19BCE248

AIM: Consider a file containing 500 integers as an input. From this file, fetch 50 integers at a time and copy them in a sorted (ascending) order. Repeat the same process for next 50 integers and merge them with previously sorted 50 integers, and hence form the sorted sequence of 100 integers.

Expected Output:- The output file must contain all 500 integers in a sorted(ascending) order.

Code:

#include<stdio.h>

#include <stdlib.h>

#include <string.h>

int \*a;

void quick\_sort(int p, int q,int n);

void swap(int \*a, int \*b);

void main(){

FILE \*fp;

fp=fopen("prac3.txt","r");

FILE \*fp1;

fp1=fopen("prac3\_out.txt","w");

char dataToBeRead[10];

int ct;

a=(int\*)malloc(sizeof(int)\*500);

int ind=0;

for (int i = 0; i < 10; i++)

{

ct=0;

while(fgets ( dataToBeRead, 50, fp ) != NULL){

if(ct==50)

break;

ct++;

a[ind++]=atoi(dataToBeRead);

}

quick\_sort( 0, ind-1,ind);

}

for(int i=0;i<500;i++)

fprintf(fp1,"%d\n",a[i]);

fclose(fp);

fclose(fp1);

}

void swap(int \*a, int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

int partion( int p, int r)

{

int pivotIndex = p + rand()%(r - p + 1);

int pivot;

int i = p - 1;

int j;

pivot = a[pivotIndex];

swap(&a[pivotIndex], &a[r]);

for (j = p; j < r; j++)

{

if (a[j] < pivot)

{

i++;

swap(&a[i], &a[j]);

}

}

swap(&a[i+1], &a[r]);

return i + 1;

}

void quick\_sort( int p, int q,int n)

{

int j;

if (p < q)

{

j = partion( p, q);

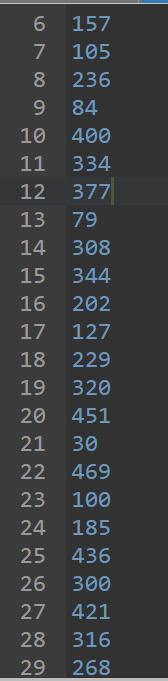
quick\_sort( p, j-1,n);

quick\_sort( j+1, q,n);

}

}

Initial File:



Final Output:-

