**Inc – header.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FILE NAME: header.h

\*

\* DESCRIPTION: all declarations of the function for various sorting tech

\*

\* DATE NAME REFERENCE REASON

\* -----------------------------------------------------------

\*Copyrights 2015 Aricent Technolegies (Holdings) ltd

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Header Files \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<stdlib.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* MACRO DECLARATIONS \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define MAX 30

#define SUCCESS 1

#define FAILURE 0

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function declarations \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int insertion\_sort(int \*,int);

void quick\_sort(int \*,int, int);

void merge\_sort(char \*\*,int, int);

**makefile:**

SRC = ./../src/

OBJ = ./../obj/

BIN = ./../bin/

GFLAGS = gcc

INC = ./../include/

CFLAGS = -c -g -Wall

OFLAGS = -lm -o

IFLAGS = -I./../include/

VFLAGS = v --tool=memcheck --leak-check=yes --show-reachable=yes

EXECS = $(BIN)final

$(BIN)final :$(OBJ)main.o $(OBJ)function.o

$(GFLAGS) $(OFLAGS)final $(OBJ)main.o $(OBJ)function.o

mv final $(BIN)

$(OBJ)main.o:$(SRC)main.c $(INC)header.h

$(GFLAGS) $(CFLAGS) $(SRC)main.c $(IFLAGS)

mv main.o $(OBJ)

$(OBJ)function.o:$(SRC)function.c $(INC)header.h

$(GFLAGS) $(CFLAGS) $(SRC)function.c $(IFLAGS)

mv function.o $(OBJ)

clean:

rm $(OBJ)\*

rm $(BIN)final

valgrind:$(EXECS)

valgrind $(VFLAGS) $(BIN)final

**src – function.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FILENAME : function.c

\* DESCRIPTION : definition of various sorting functions

\* for insertion, quick and merge sorts

\* DATE NAME REFERNCE REASON

\* --------------------------------------------------------

\*

\* Copyright @2015 Aricent

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "header.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Prototypes

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void merge(char \*\*, int, int, int, int);

static int partition(int \*, int, int);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function name: insertion\_sort

\* Descritption : sorts the given array with size n

\* Parameters : array and size of the array to sort

\* Returns : SUCCESS or FAILURE

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int insertion\_sort(int \*array,

int n)

{

int i,temp,j;

for(i=1;i<n;i++) //loop for n-1 passes

{

temp = array[i];

j = i-1;

//inserting i th element into sorted port of the array

while((temp<array[j])&&(j>=0)) // Find place to insert

{

array[j+1]=array[j];

j = j-1;

}

array[j+1]=temp;

}

return SUCCESS;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION NAME : quick\_sort

\* DESCRIPTION : It is a recursive function that partitions

\* the given list until two elements remain

\* and then does the comparision.

\* RETURN VALUE : the function return NULL.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void quick\_sort(int \*array,

int BEG,

int END)

{

int p;

if(BEG < END) /\* check at least two elements in list \*/

{

p = partition(array, BEG, END); //return the position of pivot

quick\_sort(array, BEG, p-1);

quick\_sort(array, p+1, END);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION NAME : Partition

\* DESCRIPTION : we take the 1st element as pivot and finds its

\* correct position

\* RETURN VALUE : it returns the position of the pivot element

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static int partition(int \*array, /\* array to be partitioned \*/

int BEG, /\* beginning index \*/

int END) /\* end index \*/

{

int LEFT, RIGHT, LOC, temp;

LEFT = BEG;

RIGHT = END;

LOC = BEG;

while(LEFT<=RIGHT)

{

while((array[RIGHT] >= array[LOC]) &&

(LOC != RIGHT))

{

RIGHT = RIGHT - 1;

}

if(LOC == RIGHT)

{

return LOC;

}

else

{

temp = array[LOC];

array[LOC]=array[RIGHT];

array[RIGHT]=temp;

LOC = RIGHT;

}

while((array[LEFT] <= array[LOC]) &&

(LOC != LEFT))

{

LEFT = LEFT + 1;

}

if(LOC == LEFT)

{

return LOC;

}

else

{

temp = array[LOC];

array[LOC]=array[LEFT];

array[LEFT]=temp;

LOC = LEFT;

}

}

return LOC;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION NAME : mergesort

\* DESCRIPTION : accepts array elements and divides the list till

\* the each list contains two elements using recursive

\* functions

\* RETURNS : SUCCESS or FAILURE

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void merge\_sort(char \*\*array,

int BEG,

int END)

{

int MID;

if(BEG < END) /\* check list had at least 2 elements \*/

{

MID = (BEG + END)/2;

merge\_sort(array, BEG, MID);

merge\_sort(array, MID+1, END);

merge(array, BEG, MID, MID+1, END);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTTION NAME : merge

\* DESCRIPTION : merges the elements of two sorted lists

\* RETURNS : SUCCESS or FAILURE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void merge(char \*\*array, /\* array containing data to be merged \*/

int LB1, /\* lower index of first list \*/

int UB1, /\* upper index of first list \*/

int LB2, /\* lower index of second list \*/

int UB2) /\* upper index of second list \*/

{

char c[MAX]; /\* intermediate storage \*/

int i;

int j, k;

i = LB1;

j = LB2;

k=0;

while((i <= UB1) &&

(j <= UB2))

{

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

{

c[k] = array[i];

i++;

k++;

}

else

{

c[k] = array[j];

j++;

k++;

}

}

while(i <= UB1)

{

c[k] = array[i];

k++;

i++;

}

while(j <= UB2)

{

c[k] = array[j];

k++;

j++;

}

i = LB1;

k = 0;

while(i <= UB2)

{

array[i] = c[k];

i++;

k++;

}

}

**Src – main.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* FILE NAME: main.c

\*

\* DESCRIPTION: contains the main function invoking sorting functions

\*

\* DATE NAME REFERENCE REASON

\* -------------------------------------------------------------

\*

\* Copyrights 2015 Aricent Technolegies (Holdings) ltd

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Header Files \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include"header.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FUNCTION NAME : main

\* DESCRIPTION : invokes various sorting functions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

int array[MAX];

char \*array1[MAX];

int i,n;

int choice;

printf("\n\nEnter no of numbers you want to enter:");

scanf("%d",&n);

printf("\nEnter elements:\n");

//reading the elements into the array

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

{

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

}

//displaying the array elements before sorting

printf("\nElements Before sorting :\n");

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

{

printf("%4d\n",array[i]);

}

printf("\n Please enter your choice of sorting technique:");

printf("\n 1. Insertion Sort");

printf("\n 2. Quick Sort");

printf("\n 3. Merge Sort\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

insertion\_sort(array, n);

break;

case 2:

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

break;

case 3:

printf("\n\nEnter no of numbers you want to enter:");

scanf("%d",&n);

printf("\nEnter elements:\n");

//reading the elements into the array

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

{

scanf("%s", &array1[i]);

}

merge\_sort(array1, 0, n-1);

break;

}

//displaying the array elements after sorting

printf("\n\nelements After sorting :\n");

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

{

printf("%4d\n", array[i]);

}

printf("\n");

return EXIT\_SUCCESS;

}