**LAB 2**

/\* Merge Sort - Integers \*/

#include<stdlib.h>

#include<stdio.h>

// Merges two subarrays of arr[].

// First subarray is arr[l..m]

// Second subarray is arr[m+1..r]

void merge(int arr[], int l, int m, int r)

{

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

/\* create temp arrays \*/

int L[n1], R[n2];

/\* Copy data to temp arrays L[] and R[] \*/

for (i = 0; i < n1; i++)

L[i] = arr[l + i];

for (j = 0; j < n2; j++)

R[j] = arr[m + 1+ j];

/\* Merge the temp arrays back into arr[l..r]\*/

i = 0; // Initial index of first subarray

j = 0; // Initial index of second subarray

k = l; // Initial index of merged subarray

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

/\* Copy the remaining elements of L[], if there

are any \*/

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

/\* Copy the remaining elements of R[], if there

are any \*/

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

/\* l is for left index and r is right index of the

sub-array of arr to be sorted \*/

void mergeSort(int arr[], int l, int r)

{

if (l < r)

{

// Same as (l+r)/2, but avoids overflow for

// large l and h

int m = l+(r-l)/2;

// Sort first and second halves

mergeSort(arr, l, m);

mergeSort(arr, m+1, r);

merge(arr, l, m, r);

}

}

/\* UTILITY FUNCTIONS \*/

/\* Function to print an array \*/

void printArray(int A[], int size)

{

int i;

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

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

printf("\n");

}

/\* Driver program to test above functions \*/

int main()

{

int n;

// int arr[] = {12, 11, 13, 5, 6, 7};

int arr[100];

// int arr\_size = sizeof(arr)/sizeof(arr[0]);

printf("No. of Elements\n");

scanf("%d", &n);

printf("Elements\n");

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

{

/\* code \*/

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

}

printf("Given array is \n");

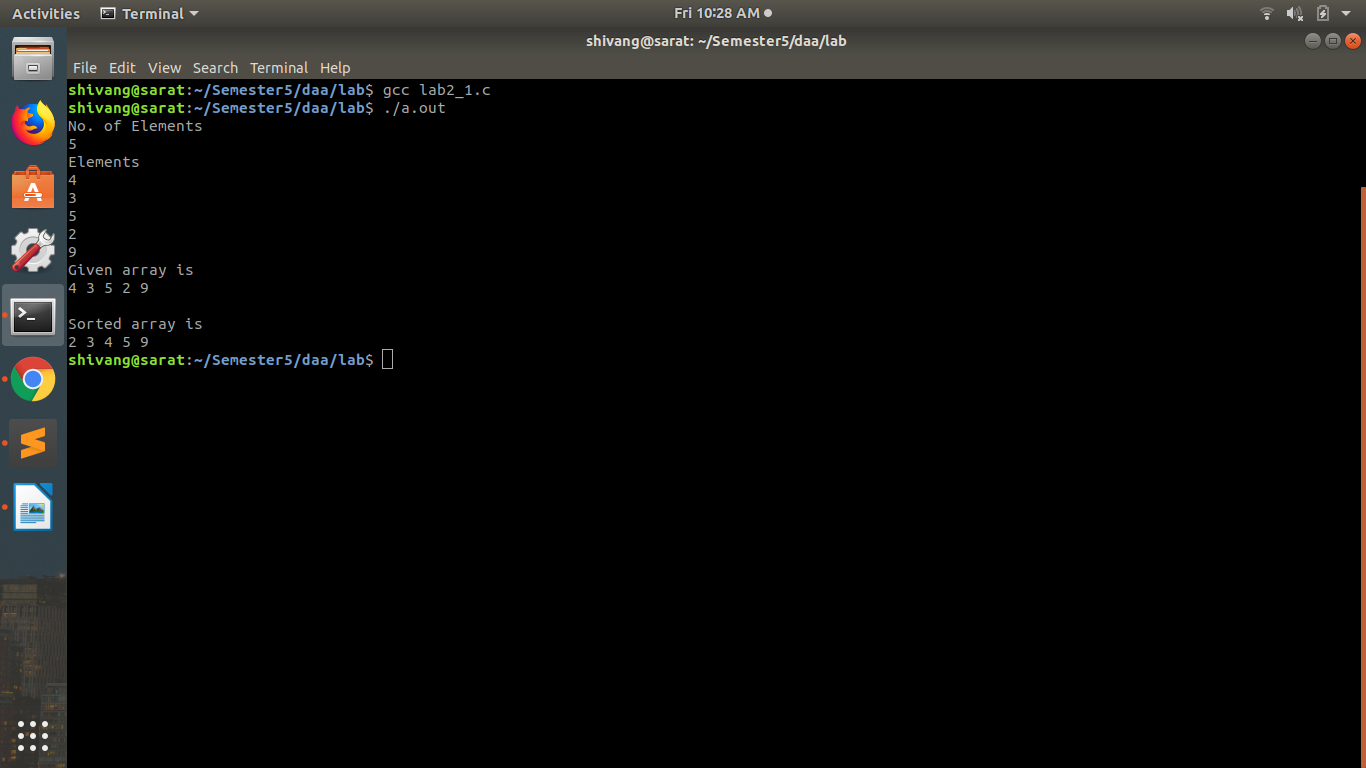
printArray(arr, n);

mergeSort(arr, 0, n - 1);

printf("\nSorted array is \n");

printArray(arr, n);

return 0;

}

/\* Merge Sord -Characters \*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h> //To use the string functions like strcmp and strcpy

#define MAX 10 // This is the default size of every string

void Merge(char\* arr[],int low,int mid,int high) //Merging the Array Function

{

int nL= mid-low+1;

int nR= high-mid;

char\*\* L=malloc(sizeof(char \*)\*nL);

char\*\* R=malloc(sizeof(char \*)\*nR);

int i;

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

{

L[i]=malloc(sizeof(arr[low+i]));

strcpy(L[i],arr[low+i]);

}

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

{

R[i]=malloc(sizeof(arr[mid+i+1]));

strcpy(R[i],arr[mid+i+1]);

}

int j=0,k;

i=0;

k=low;

while(i<nL&&j<nR)

{

if(strcmp(L[i],R[j])<0)strcpy(arr[k++],L[i++]);

else strcpy(arr[k++],R[j++]);

}

while(i<nL)strcpy(arr[k++],L[i++]);

while(j<nR)strcpy(arr[k++],R[j++]);

}

void MergeSort(char\* arr[],int low,int high) //Main MergeSort function

{

if(low<high)

{

int mid=(low+high)/2;

MergeSort(arr,low,mid);

MergeSort(arr,mid+1,high);

Merge(arr,low,mid,high);

}

}

int main()

{

printf("\nEnter the size of the array desired: ");

int size; //This is the String array size

scanf("%d",&size);

char\*\* arr= malloc(sizeof(char \*)\* size); //Creating required string array

printf("\nEnter the strings of the array: ");

int i;

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

{

arr[i]=malloc(sizeof(char)\*MAX);

printf("\nEnter String: ");

scanf("%s",arr[i]);

}

MergeSort(arr,0,size-1);

printf("\nThe Sorted Array is: \n");

for(i=0;i<size;i++)printf("%s\n",arr[i]);

return 0;

}

