DAA–Lab3–January25

Vishal Gauba
1410110501

Source Code:

```
#include"stdio.h"
#include"stdlib.h"
#include"math.h"
#include"time.h"
#include"assert.h"

int result[100000];

void bubble_sort(int* array, int n){
        printf("Bubbling the array...\n");
        int c,d,swap;
        for(c=0;c<(n-1);c++){
                for(d=0;d<n-c-1;d++){
          if(array[d]>array[d+1]){
                                swap   =array[d];
                                array[d] =array[d+1];
                                array[d+1]=swap;
                        }
                }
        }
}

int largest(int a[], int n){
 int large=a[0], i;
 for(i=1;i<n;i++)
 {
  if(large<a[i])
    large=a[i];
 }
 return large;
}

void radix_sort(int *a, int n){
        int i, b[100000], m=0, exp=1;
        for(i=0;i<n;i++){
```

```
                if(a[i]>m)
                 m=a[i];
        }
        while(m/exp>0){
                int box[10]={0};
                for(i=0;i<n;i++)
                 box[a[i]/exp%10]++;
                for(i=1;i<10;i++)
                 box[i]+=box[i-1];
                for(i=n-1;i>=0;i--)
                 b[--box[a[i]/exp%10]]=a[i];
                for(i=0;i<n;i++)
                 a[i]=b[i];
                exp*=10;
                #ifdef SHOWPASS
                 printf("\n\nPASS :");
                print(a,n);
                #endif
        }
}

void merge(int arr[], int start1, int end1, int start2, int end2){
 int i,j,k;
 k=start1;
 i=start1;
 j=start2;

 //while there are elements in both lists, keep picking the smaller one
 while((i<=end1)&&(j<=end2))
 {
  if(arr[i]<=arr[j])
  {
   result[k]=arr[i];
   i++;
   k++;
  }
  else
  {
   result[k]=arr[j];
   j++;
   k++;
  }
 }

 //one of the lists would have been finished. append the other list at end
 //Remaining part of List 1
 while(i<=end1)
 {
```

```
 result[k] = arr[i];
 i++;
 k++;
}
// Remaining part of List 2
while (j <= end2)
{
 result[k] = arr[j];
 j++;
 k++;
}

assert(i==end1+1);
assert(j==end2+1);
assert(k==end2+1);

// Copy the merged list back to the input array
for(i=start1; i<=end2; i++)
 arr[i] = result[i];
}

void recursive_mergesort( int arr[], int start, int end){
 int mid;
 int n; // number of items
 n = end-start+1;
 mid = (start+end)/2;


 if (n < 2)
  return;

 // recursive_sort each part. merge the two sorted parts
 recursive_mergesort(arr, start, mid);
 recursive_mergesort(arr, mid+1, end);

 // merge the two sorted parts
 merge(arr, start, mid, mid+1, end);
}

Int main (int argc, char const *argv[])
{
        clock_t t0 = clock();
        int n=10000, i, *array = (int*) malloc(n*sizeof(int));

        printf("Generating the array...\n");
        for (i=0; i<n; ++i){
                array[i] = rand()%n;
        }
```

```
        //bubble_sort(array,n);
        radix_sort(array,n); /*change 2D bucket array size in the function too*/
        //recursive_mergesort(array,0,n-1);/*change global variable result too*/

        //for(i=0;i<n;++i){
        //  printf("%d\n",array[i]);
        //}
        clock_t t1=clock();
        printf("Time taken to execute=%lf\n",(double)(t1-t0)/CLOCKS_PER_SEC);
        return 0;
}
```

Analysis Table:

Time is in seconds.

Input Size	Bubble Sort time	Merge Sort time	Radix Sort time
10^3	00.015625	0	0
10^4	00.265625	00.015625	00.015625
10^5	30.531250	00.234375	00.125000