Name : Rahul Thapar

ID : 1410110321

Write three functions separately that the main function calls one after another, where each function sorts the same array using a different algorithm, namely:

1. Radix sort

2. Bubble sort

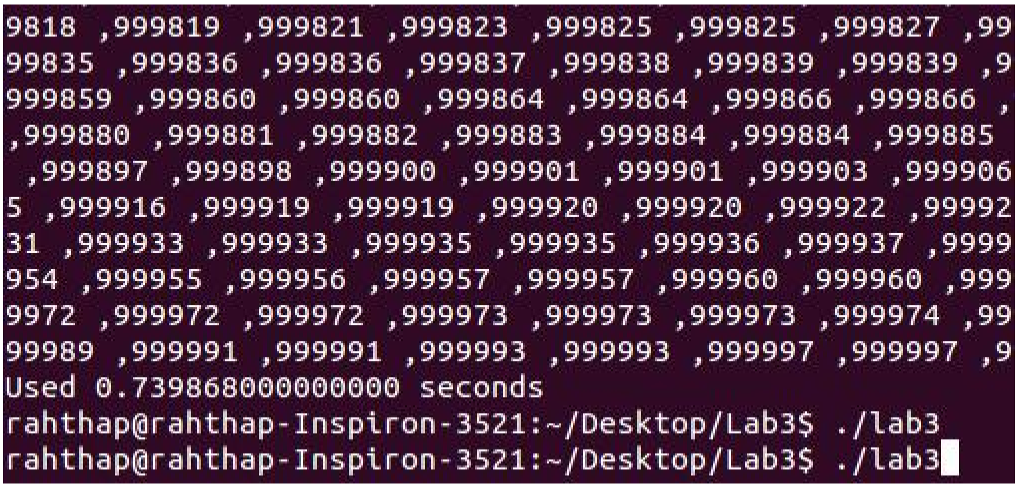
3. Merge sort

Compute the average run time for each of the three techniques separately, for n = 10^3, 10^6

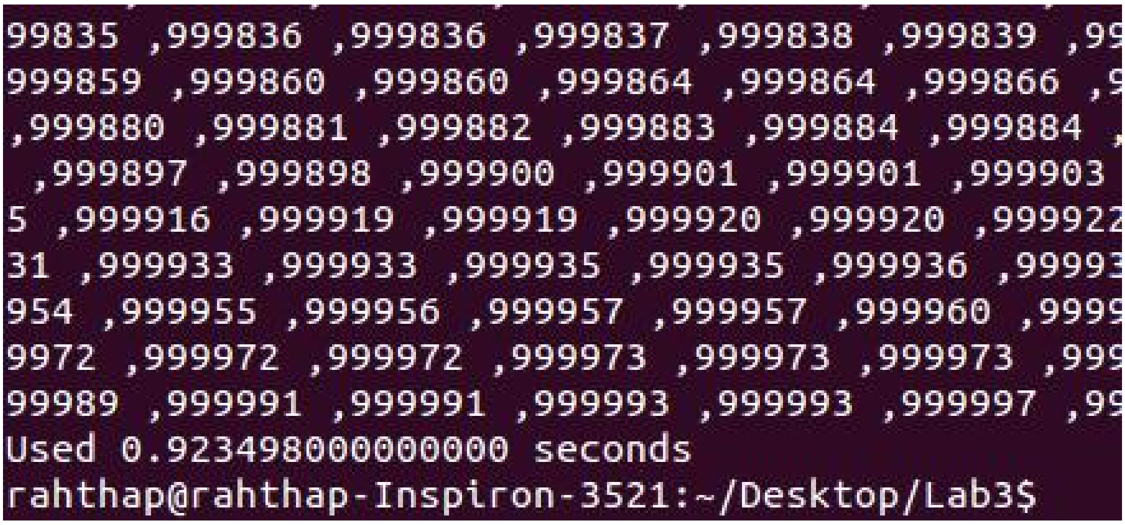
Solution :

S. No	Sorting Algorithm	Data Set (value of n)	Time Taken (seconds)
1.	Radix Sort	10^3	0.000818000000000
2.	Bubble Sort	10^3	0.002820000000000
3.	Merge Sort	10^3	0.000321000000000

S. No	Sorting Algorithm	Data Set (value of n)	Time Taken (seconds)
1.	Radix Sort	10^6	0.739868000000000
2.	Bubble Sort	10^6	Unable to process
3.	Merge Sort	10^6	0.923498000000000



```
9818 ,999819 ,999821 ,999823 ,999825 ,999825 ,999827 ,99
99835 ,999836 ,999836 ,999837 ,999838 ,999839 ,999839 ,9
999859 ,999860 ,999860 ,999864 ,999864 ,999866 ,999866 ,
,999880 ,999881 ,999882 ,999883 ,999884 ,999884 ,999885
 ,999897 ,999898 ,999900 ,999901 ,999901 ,999903 ,999906
5 ,999916 ,999919 ,999919 ,999920 ,999920 ,999922 ,99992
31 ,999933 ,999933 ,999935 ,999935 ,999936 ,999937 ,9999
954 ,999955 ,999956 ,999957 ,999957 ,999960 ,999960 ,999
9972 ,999972 ,999972 ,999973 ,999973 ,999973 ,999974 ,99
99989 ,999991 ,999991 ,999993 ,999993 ,999997 ,999997 ,9
Used 0.739868000000000 seconds
rahthap@rahthap-Inspiron-3521:~/Desktop/Lab3$ ./lab3
rahthap@rahthap-Inspiron-3521:~/Desktop/Lab3$ ./lab3
```



```
99835 ,999836 ,999836 ,999837 ,999838 ,999839 ,99
999859 ,999860 ,999860 ,999864 ,999864 ,999866 ,9
,999880 ,999881 ,999882 ,999883 ,999884 ,999884 ,
 ,999897 ,999898 ,999900 ,999901 ,999901 ,999903
5 ,999916 ,999919 ,999919 ,999920 ,999920 ,99992
31 ,999933 ,999933 ,999935 ,999935 ,999936 ,9999
954 ,999955 ,999956 ,999957 ,999957 ,999960 ,9999
9972 ,999972 ,999972 ,999973 ,999973 ,999973 ,999
99989 ,999991 ,999991 ,999993 ,999993 ,999997 ,99
Used 0.923498000000000 seconds
rahthap@rahthap-Inspiron-3521:~/Desktop/Lab3$
```

CODE :

```
/***

*

* @author : Rahul Thapar

* Date : 25 January 2017

* ID : 1410110321

*

***/



#include<stdio.h>
```

```
#include<string.h>
#include<time.h>
#include<unistd.h>
#define MAX 1000000

void printArray(int * array, int size){

   int i;
   printf("{ ");
   for (i = 0; i < size; i++)
      printf("%d ,", array[i]);
   printf("}\n");
}

int findlargest_Number(int * array, int size){
   int i;
   int largest_Number = -1;
   for(i = 0; i < size; i++){
      if(array[i] > largest_Number)
         largest_Number = array[i];
   }
   return largest_Number;
}
```

```
void partition(int arr[],int lower_value,int higher_value){

    int mid;

    if(lower_value<higher_value){

         mid=(lower_value+higher_value)/2;

         partition(arr,lower_value,mid);

         partition(arr,mid+1,higher_value);

         mergeSort(arr,lower_value,mid,higher_value);

    }

}


// RADIX SORT

radixSort(int * array, int size){


  printf("\n\nRunning Radix Sort .........!\n\n");

  int i;

  int semiSorted[size];

  int significant_Digit = 1;

  int largest_Number = findlargest_Number(array, size);


  while (largest_Number / significant_Digit > 0){


    int bucket[10] = { 0 };


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

```
         bucket[(array[i] / significant_Digit) % 10]++;

      for (i = 1; i < 10; i++)
         bucket[i] += bucket[i - 1];

      for (i = size - 1; i >= 0; i--)
         semiSorted[--bucket[(array[i] / significant_Digit) % 10]] =
array[i];



      for (i = 0; i < size; i++)
         array[i] = semiSorted[i];

      significant_Digit *= 10;
   }
   printArray(array, size);
}

// BUBBLE SORT
bubbleSort(int *array,int size){
   printf("\n\nRunning bubble Sort .........!\n\n");
   int c,d,swap;
   for (c = 0 ; c < ( size - 1 ); c++)
   {
    for (d = 0 ; d < size - c - 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use < */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
  printArray(array,size);
}

// MERGE SORT
mergeSort(int arr[],int lower_value,int mid,int higher_value){
    int i,m,k,l,flag[MAX];

    l=lower_value;
    i=lower_value;
    m=mid+1;
    while((l<=mid)&&(m<=higher_value)){
         if(arr[l]<=arr[m]){
             flag[i]=arr[l];
             l++;
         }
         else{
             flag[i]=arr[m];
             m++;
         }
         i++;
    }
    if(l>mid){
         for(k=m;k<=higher_value;k++){
             flag[i]=arr[k];
             i++;
         }
    }
    else{
         for(k=l;k<=mid;k++){
             flag[i]=arr[k];
             i++;
         }
    }
    for(k=lower_value;k<=higher_value;k++){
         arr[k]=flag[k];
    }
}
```

```
void main()

{

   int array[MAX];

   int i,n=0;

   int size;



   clock_t start, stop;



         //FILE* f1 = fopen("Random.txt", "r");   // Corresponds to data
set of n = 10^3

         FILE* f1 = fopen("Random_1.txt", "r");   // Corresponds to data
set of n = 10^6

         i=0;

         while( fscanf(f1, "%d,", &n) > 0 ) // parse %d follower_valueed
by ','

            {

                  array[i++] = n;

            }



   start = clock();

   radixSort(array,MAX);

   stop = clock();

   printf("Used %0.15f seconds \n",
(double)(stop-start)/CLOCKS_PER_SEC);
```

```
  start = clock();

  bubbleSort(array,MAX);

  stop = clock();

  printf("Used %0.15f seconds \n",
(double)(stop-start)/CLOCKS_PER_SEC);





  //mergeSort(array,MAX);

  start = clock();

  partition(array,0,MAX-1);

  printf("\n\nRunning Merge Sort .........!\n\n");

 printArray(array,MAX);

 stop = clock();

  printf("Used %0.15f seconds \n",
(double)(stop-start)/CLOCKS_PER_SEC);



}
```