LAB 3

Sushant Bansal
1410110454
25th January, 2016

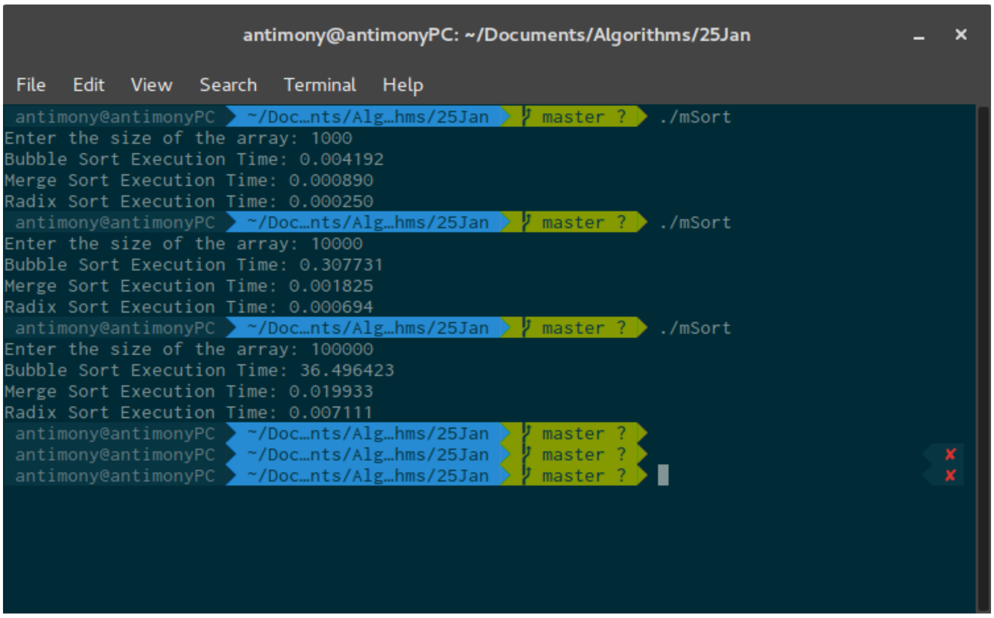
CODE

```
/*
Calculate time based on various sorting algorithms
Sushant Bansal
1410110454
25th January 2016
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_LEN 10000000
#define SHOWPASS

void printArray(int *arr, int len);
void copyArray(int *arrOrigin,int *copy, int len);
void radix_sort(int *a,int n);
void bubble_sort(int *arr,int len);
void merge_sort(int low,int high);
void merge(int low, int mid, int high);

int radixSortArray[MAX_LEN];
int mergeSortArray[MAX_LEN];
int bubbleSortArray[MAX_LEN];
int tempArr[MAX_LEN];

int main(){
       clock_t begin,end;
       double time_spent;
       int numOfDigits = 10;
       printf("Enter the size of the array: ");
       scanf("%d", &numOfDigits);
       srand(time(NULL));
       for (int i = 0; i < numOfDigits; i++)
       {
              radixSortArray[i] = rand()%999 ;
       }
       copyArray(radixSortArray,mergeSortArray,numOfDigits);
       copyArray(radixSortArray,bubbleSortArray,numOfDigits);

       begin = clock();
       bubble_sort(bubbleSortArray,numOfDigits);
       end = clock();
       //printArray(bubbleSortArray,numOfDigits);
       time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
       printf("Bubble Sort Execution Time: %f\n",time_spent);

       begin = clock();
       merge_sort(0,numOfDigits);
       end = clock();
       // printArray(mergeSortArray,numOfDigits-1);
       time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
       printf(”Merge Sort Execution Time: %f\n”,time_spent);

       begin = clock();
       radix_sort(radixSortArray,numOfDigits);
       end = clock();
       // printArray(radixSortArray,numOfDigits);
       time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
       printf(”Radix Sort Execution Time: %f\n”,time_spent);

}

void printArray(int *arr, int len){
       for(int i = 0;i < len ;i++)
              printf(”%d\n”,arr[i]);
}

void copyArray(int *arrOrigin,int *copy, int len){
       for(int i = 0;i < len ;i++)
              copy[i] = arrOrigin[i];
}

void merge_sort(int low,int high){
       int mid;

       if(low < high) {
      mid = (low + high) / 2;
      merge_sort(low, mid);
      merge_sort(mid+1, high);
      merge(low, mid, high);
       }
       else {
      return;
       }
}

void merge(int low, int mid, int high){

       int l1, l2, i;
       int *a = mergeSortArray;
       int *b = tempArr;
       for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
           if(a[l1] <= a[l2])
               b[i] = a[l1++];
           else
               b[i] = a[l2++];
          }

          while(l1 <= mid)
             b[i++] = a[l1++];

          while(l2 <= high)
             b[i++] = a[l2++];

          for(i = low; i <= high; i++)
             a[i] = b[i];
}

void bubble_sort(int *arr,int len){
       int swap;
       for(int i =0;i<len;i++){
              for(int j=0;j<len-1-i;j++){
                     if(arr[j] > arr[j+1]) {
            swap = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = swap;
              }
```

```
                }
        }
}

void radix_sort(int *a, int n) {
        int i, m = 0, exp = 1;
        int *b = tempArr;
        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;
        }
}
```

Observations

	N = 1000	**N = 10000**	**N = 100000**
Bubble Sort	0.004192	0.307731	36.496423
Merge Sort	0.00089	0.001825	0.019933
Radix Sort	0.00025	0.000694	0.007111



```
 antimony@antimonyPC  ~/Doc…nts/Alg…hms/25Jan   master ?  ./mSort
Enter the size of the array: 1000
Bubble Sort Execution Time: 0.004192
Merge Sort Execution Time: 0.000890
Radix Sort Execution Time: 0.000250
 antimony@antimonyPC  ~/Doc…nts/Alg…hms/25Jan   master ?  ./mSort
Enter the size of the array: 10000
Bubble Sort Execution Time: 0.307731
Merge Sort Execution Time: 0.001825
Radix Sort Execution Time: 0.000694
 antimony@antimonyPC  ~/Doc…nts/Alg…hms/25Jan   master ?  ./mSort
Enter the size of the array: 100000
Bubble Sort Execution Time: 36.496423
Merge Sort Execution Time: 0.019933
Radix Sort Execution Time: 0.007111
 antimony@antimonyPC  ~/Doc…nts/Alg…hms/25Jan   master ?
 antimony@antimonyPC  ~/Doc…nts/Alg…hms/25Jan   master ?                                       ✘
 antimony@antimonyPC  ~/Doc…nts/Alg…hms/25Jan   master ?                                       ✘
```