Data Structure Assignment - 4

Problems

- Write a C program to sort the given array element using Bubble Sort Algorithm
- Write a C program to sort the given array element using Insertion Sort Algorithm
- Write a C program to sort the given array element using Selection Sort Algorithm
- Write a C program to sort the given array element using Merge Sort Algorithm
- Write a C program to sort the given array element using Quick Sort Algorithm
- Write a C program to sort the given array element using Radix Sort Algorithm

Problem - 1

C Code

```
/**
* Data Structures Assignment - 4
* CS 392
* Problem 1:
*       Write a C program to sort the given array element using Bubble Sort Algorithm
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/

/**
*   Algorithm:
*       begin BubbleSort(list)
*
*        for all elements of list
*            if list[i] > list[i+1]
*              swap(list[i], list[i+1])
*            end if
*          end for
*
*          return list
*
*        end BubbleSort
*
**/


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

int *array;
int len;

void init() {
  scanf("%d",&len);
  array = malloc(sizeof(int)*len);
  for (int i=0; i<len; i++) {
```

```
    scanf("%d",(array+i));
  }
}

void show() {
  for(int i=0; i < len; i++)
      printf("%d ",*(array + i));
  printf("\n");
}

void BubbleSort() {
  int temp;

  for (int pass = len-1; pass > 0; pass--) {
    for (int i=0; i < pass; i++) {
      if(array[i] > array[i+1]) {
        temp = array[i];
        array[i] = array[i+1];
        array[i+1] = temp;
      }
    }
  }
}

int main(int argc, char* argv[]) {
  // For testing puposes no prefix text will be given for input
  init();

  BubbleSort();
  if(argc < 2)
    show();

  return 0;
}


```

Output

Length of array: 10

Sorted Input:

```
➜ time ./4_1 < sorted-input
0 1 2 3 4 5 6 7 8 9
./4_1 < sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Reverse Sorted Input:

```
➜ time ./4_1 < rev_sorted-input
0 1 2 3 4 5 6 7 8 9
./4_1 < rev_sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Randomized Input:

```
➜ time ./4_1 < rand-input
0 2 4 4 6 6 7 7 7 7
./4_1 < rand-input  0.00s user 0.00s system 0% cpu 0.002 total
```

Length of array: 100000

Sorted Input:

```
➜ time ./4_1 --no < sorted-input
./4_1 --no < sorted-input  13.14s user 0.00s system 99% cpu 13.146 total
```

Reverse Sorted Input:

```
➜ time ./4_1 --no <` rev_sorted-input
./4_1 --no < rev_sorted-input  20.44s user 0.00s system 99% cpu 20.449 total
```

Randomized Input:

```
➜ time ./4_1 --no < rand-input
./4_1 --no < rand-input  31.52s user 0.00s system 99% cpu 31.527 total
```

Problem - 2

C Code

```
/**
* Data Structures Assignment - 4
* CS 392
* Problem 2:
*       Write a C program to sort the given array element using Selection Sort Algorithm
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/
```

```

/**
*   Algorithm:
*       begin SelectionSort(list)
*
*        for element e in list:
*            set min = indexof(e)
*            for element f in list starting from index min:
*              if f < e:
*                set min = indexof(f)
*
*            swap elements e and min
*
*          end for
*
*          return list
*
*        end BubbleSort
*
**/


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

int *array;
int len;

void init() {
  scanf("%d",&len);
  array = malloc(sizeof(int)*len);
  for (int i=0; i<len; i++) {
    scanf("%d",(array+i));
  }
}

void show() {
  for(int i=0; i < len; i++)
      printf("%d ",*(array + i));
  printf("\n");
}

void SelectionSort() {
  int min, temp;

  for (int i=0; i < len - 1; i++) {
    min = i;
    for (int j = i + 1; j < len; j++) {
        if(array[j] < array[min])
          min = j;
    }
    // swap
    temp = array[min];
    array[min] = array[i];
    array[i] = temp;
  }

}

int main(int argc, char* argv[]) {
  // For testing puposes no prefix text will be given for input
```

```
  init();

  SelectionSort();

  if(argc < 2)
    show();

  return 0;
}


```

Output

Length of array: 10

Sorted Input:

```
➜ time ./4_2 < sorted-input
0 1 2 3 4 5 6 7 8 9
./4_2 < sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Reverse Sorted Input:

```
➜ time ./4_2 < rev_sorted-input
0 1 2 3 4 5 6 7 8 9
./4_2 < rev_sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Randomized Input:

```
➜ time ./4_2 < rand-input
0 2 4 4 6 6 7 7 7 7
./4_2 < rand-input  0.00s user 0.00s system 0% cpu 0.002 total
```

Length of array: 100000

Sorted Input:

```
➜ time ./4_2 --no < sorted-input
./4_2 --no < sorted-input  10.72s user 0.00s system 99% cpu 10.725 total
```

Reverse Sorted Input:

```
➜ time ./4_2 --no < rev_sorted-input
./4_2 --no < rev_sorted-input  12.92s user 0.00s system 99% cpu 12.931 total
```

Randomized Input:

```
➜ time ./4_2 --no < rand-input
./4_2 --no < rand-input  10.74s user 0.00s system 99% cpu 10.770 total
```

Problem - 3

C Code

```
/**
* Data Structures Assignment - 4
* CS 392
* Problem 3:
*       Write a C program to sort the given array element using Insertion Sort Algorithm
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/


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

int *array;
int len;

void init() {
  scanf("%d",&len);
  array = malloc(sizeof(int)*len);
  for (int i=0; i<len; i++) {
    scanf("%d",(array+i));
  }
}

void show() {
  for(int i=0; i < len; i++)
      printf("%d ",*(array + i));
  printf("\n");
}

void InsertionSort() {
  int temp;
  int j;
  for (int i=0; i < len; i++) {
    temp = array[i];
    for (j = i;j>0 && array[j-1] > temp;j--) {
      array[j] = array[j-1];
    }
    array[j] = temp;
  }
}

int main(int argc, char* argv[]) {
  // For testing puposes no prefix text will be given for input
  init();

  InsertionSort();
  if(argc < 2)
    show();

  return 0;
}

```

```
57
```

Output

Length of array: 10

Sorted Input:

```
➜ time ./4_3 < sorted-input
0 1 2 3 4 5 6 7 8 9
./4_3 < sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Reverse Sorted Input:

```
➜ time ./4_3 < rev_sorted-input
0 1 2 3 4 5 6 7 8 9
./4_3 < rev_sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Randomized Input:

```
➜ time ./4_3 < rand-input
0 2 4 4 6 6 7 7 7 7
./4_3 < rand-input  0.00s user 0.00s system 0% cpu 0.002 total
```

Length of array: 100000

Sorted Input:

```
➜ time ./4_3 --no < sorted-input
./4_3 --no < sorted-input  0.03s user 0.00s system 98% cpu 0.033 total
```

Reverse Sorted Input:

```
➜ time ./4_3 --no < rev_sorted-input
./4_3 --no < rev_sorted-input  15.92s user 0.00s system 99% cpu 15.926 total
```

Randomized Input:

```
➜ time ./4_3 --no < rand-input
./4_3 --no < rand-input  8.00s user 0.00s system 99% cpu 8.004 total
```

Problem - 4

C Code

```
/**
* Data Structures Assignment - 4
* CS 392
* Problem 4:
*       Write a C program to sort the given array element using Merge Sort Algorithm
```

```
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/



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

int *array;
int *temp;
int len;

void Merge(int left, int mid, int right);

void init() {
  scanf("%d",&len);
  array = malloc(sizeof(int)*len);
  temp = malloc(sizeof(int)*len);

  for (int i=0; i<len; i++) {
    scanf("%d",(array+i));
  }
}

void show() {
  for(int i=0; i < len; i++)
      printf("%d ",*(array + i));
  printf("\n");
}

void MergeSort(int left, int right) {
  int mid;
  if (right > left) {
    mid = (right + left) / 2;
    MergeSort(left, mid);
    MergeSort(mid + 1, right);
    Merge(left, mid+1, right);
  }
}
void Merge(int left, int mid, int right) {
  int i, left_end, size, temp_pos;

  left_end = mid -1;
  temp_pos = left;

  size = right - left + 1;

  while((left <= left_end) && (mid <= right)) {
    if(array[left] <= array[mid])
      temp[temp_pos++] = array[left++];
    else
      temp[temp_pos++] = array[mid++];
  }
  while( left <= left_end) {
    temp[temp_pos++] = array[left++];
  }
  while( mid <= right) {
```

```
    temp[temp_pos++] = array[mid++];
  }
  for(int i=0; i<= size; i++) {
    array[right] = temp[right--];
  }
}

int main(int argc, char* argv[]) {
  // For testing puposes no prefix text will be given for input
  init();

  MergeSort(0,len-1);
  if(argc < 2)
    show();

  return 0;
}


```

Output

Length of array: 10

Sorted Input:

```
➜ time ./4_4 < sorted-input
0 1 2 3 4 5 6 7 8 9
./4_4 < sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Reverse Sorted Input:

```
➜ time ./4_4 < rev_sorted-input
0 1 2 3 4 5 6 7 8 9
./4_4 < rev_sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Randomized Input:

```
➜ time ./4_4 < rand-input
0 2 4 4 6 6 7 7 7 7
./4_4 < rand-input  0.00s user 0.00s system 0% cpu 0.002 total
```

Length of array: 100000

Sorted Input:

```
➜ time ./4_4 --no < sorted-input
./4_4 --no < sorted-input  0.02s user 0.00s system 97% cpu 0.025 total
```

Reverse Sorted Input:

```
➜ time ./4_4 --no < rev_sorted-input
./4_4 --no < rev_sorted-input  0.04s user 0.00s system 98% cpu 0.045 total
```

Randomized Input:

```
➜ time ./4_4 --no < rand-input
./4_4 --no < rand-input  0.04s user 0.00s system 97% cpu 0.037 total
```

Problem - 5

C Code

```
/**
* Data Structures Assignment - 4
* CS 392
* Problem 5:
*       Write a C program to sort the given array element using Quick Sort Algorithm
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/




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

int *array;
int len;

void init() {
  scanf("%d",&len);
  array = malloc(sizeof(int)*len);
  for (int i=0; i<len; i++) {
    scanf("%d",(array+i));
  }
}

void show() {
  for(int i=0; i < len; i++)
      printf("%d ",*(array + i));
  printf("\n");
}

void QuickSort(int low, int high) {
```

```
  int pos_pivot = low;
  int pivot = array[low];
  int left = low;
  int right = high;
  int temp = 0;
  // Base case for recursion
  if(high > low) {
    while (right > left) {
      while(array[left] <= pivot)
        left++;
      while(array[right] > pivot)
        right--;

      if(left < right) {
        temp = array[right];
        array[right] = array[left];
        array[left] = temp;
      }
    }

    array[pos_pivot] = array[right];
    array[right] = pivot;
    pos_pivot = right;


    QuickSort(low, pos_pivot-1);
    QuickSort(pos_pivot+1, high);
  }
  return;
}

int main(int argc, char* argv[]) {
  // For testing puposes no prefix text will be given for input
  init();

  QuickSort(0,len-1);
  if(argc < 2)
    show();

  return 0;
}


```

Output

Length of array: 10

Sorted Input:

```
➜ time ./4_5 < sorted-input
0 1 2 3 4 5 6 7 8 9
./4_5 < sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Reverse Sorted Input:

```
➜ time ./4_5 < rev_sorted-input
0 1 2 3 4 5 6 7 8 9
./4_5 < rev_sorted-input  0.00s user 0.00s system 0% cpu 0.003 total
```

Randomized Input:

```
➜ time ./4_5 < rand-input
0 2 4 4 6 6 7 7 7 7
./4_5 < rand-input  0.00s user 0.00s system 0% cpu 0.002 total
```

Length of array: 100000

Sorted Input:

```
➜ time ./4_5 --no < sorted-input
./4_5 --no < sorted-input  8.80s user 0.00s system 99% cpu 8.811 total
```

Reverse Sorted Input:

```
➜ time ./4_5 --no < rev_sorted-input
./4_5 --no < rev_sorted-input  8.76s user 0.00s system 99% cpu 8.768 total
```

Randomized Input:

```
➜ time ./4_5 --no < rand-input
./4_5 --no < rand-input  0.04s user 0.00s system 96% cpu 0.045 total
```

Problem - 6

C Code

```
1
2  /**
3  * Data Structures Assignment - 4
4  * CS 392
5  * Problem 6:
6  *       Write a C program to sort the given array element using Radix Sort Algorithm
7  *
8  * Joydeep Mukherjee
9  * CSE,3rd Sem, 11000116030
10 * GCETTS 2017
11 *
12 **/
13
```

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

int* array;
int len;

void init() {
  scanf("%d",&len);
  array = malloc(sizeof(int)*len);
  for (int i=0; i<len; i++) {
    scanf("%d",(array+i));
  }
}

void show() {
  for(int i=0; i < len; i++)
      printf("%d ",*(array + i));
  printf("\n");
}

int getMax(int n) {
  int mx = array[0];
  for(int i=0; i < n; i++) {
    if (array[i] > mx)
      mx = array[i];
  }
  return mx;
}
void countSort(int n, int ex) {
  int* output = malloc(sizeof(int)*n);
  int i,count[10] = {0};

  for (i = 0; i < n; i++)
    count[ (array[i]/ex)%10 ]++;

  for (i = 1; i < n; i++)
    count[i] += count[i-1];

  for (i = n - 1; i>= 0; i-- ) {
    output[count[ (array[i]/ex)%10 ] - 1] = array[i];
    count[ (array[i]/ex)%10 ]--;
  }

  for (i = 0; i < n; i++)
    array[i] = output[i];
}

void radixsort(int n) {
  int m = getMax(n);

  for (int ex = 1;m/ex > 0; ex *= 10)
    countSort(n,ex);
}

int main(int argc, char* argv[]) {
  // For testing puposes no prefix text will be given for input
  init();

  radixsort(len);
  if(argc < 2)
    show();
```

```
75
76   return 0;
77 }
78
```