

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

```
1  /**
2  * Data Structures Assignment - 4
3  * CS 392
4  * Problem 1:
5  *     Write a C program to sort the given array element using Bubble Sort Algorithm
6  *
7  * Joydeep Mukherjee
8  * CSE, 3rd Sem, 11000116030
9  * GCETTS 2017
10 *
11 **/
12
13 /**
14 * Algorithm:
15 *     begin BubbleSort(list)
16 *
17 *     for all elements of list
18 *         if list[i] > list[i+1]
19 *             swap(list[i], list[i+1])
20 *         end if
21 *     end for
22 *
23 *     return list
24 *
25 *     end BubbleSort
26 *
27 **/
28
29
30 #include <stdio.h>
31 #include <stdlib.h>
32
33 int *array;
34 int len;
35
36 void init() {
37     scanf("%d",&len);
38     array = malloc(sizeof(int)*len);
39     for (int i=0; i<len; i++) {
```

```
40     scanf("%d", (array+i));
41 }
42 }
43
44 void show() {
45     for(int i=0; i < len; i++)
46         printf("%d ", *(array + i));
47     printf("\n");
48 }
49
50 void BubbleSort() {
51     int temp;
52
53     for (int pass = len-1; pass > 0; pass--) {
54         for (int i=0; i < pass; i++) {
55             if(array[i] > array[i+1]) {
56                 temp = array[i];
57                 array[i] = array[i+1];
58                 array[i+1] = temp;
59             }
60         }
61     }
62 }
63
64 int main(int argc, char* argv[]) {
65     // For testing puposes no prefix text will be given for input
66     init();
67
68     BubbleSort();
69     if(argc < 2)
70         show();
71
72     return 0;
73 }
74
75
```

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**

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

```

12
13 /**
14 *   Algorithm:
15 *       begin SelectionSort(list)
16 *
17 *           for element e in list:
18 *               set min = indexof(e)
19 *               for element f in list starting from index min:
20 *                   if f < e:
21 *                       set min = indexof(f)
22 *
23 *               swap elements e and min
24 *
25 *           end for
26 *
27 *       return list
28 *
29 *   end BubbleSort
30 *
31 **/
32
33
34 #include <stdio.h>
35 #include <stdlib.h>
36
37 int *array;
38 int len;
39
40 void init() {
41     scanf("%d",&len);
42     array = malloc(sizeof(int)*len);
43     for (int i=0; i<len; i++) {
44         scanf("%d",(array+i));
45     }
46 }
47
48 void show() {
49     for(int i=0; i < len; i++)
50         printf("%d ",*(array + i));
51     printf("\n");
52 }
53
54 void SelectionSort() {
55     int min, temp;
56
57     for (int i=0; i < len - 1; i++) {
58         min = i;
59         for (int j = i + 1; j < len; j++) {
60             if(array[j] < array[min])
61                 min = j;
62         }
63         // swap
64         temp = array[min];
65         array[min] = array[i];
66         array[i] = temp;
67     }
68
69 }
70
71 int main(int argc, char* argv[]) {
72     // For testing puposes no prefix text will be given for input

```

```
73  init();
74
75  SelectionSort();
76
77  if(argc < 2)
78      show();
79
80  return 0;
81 }
82
83
```

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

```
1  /**
2  * Data Structures Assignment - 4
3  * CS 392
4  * Problem 3:
5  *     Write a C program to sort the given array element using Insertion Sort Algorithm
6  *
7  * Joydeep Mukherjee
8  * CSE,3rd Sem, 11000116030
9  * GCETTS 2017
10 *
11 **/
12
13
14 #include <stdio.h>
15 #include <stdlib.h>
16
17 int *array;
18 int len;
19
20 void init() {
21     scanf("%d",&len);
22     array = malloc(sizeof(int)*len);
23     for (int i=0; i<len; i++) {
24         scanf("%d",(array+i));
25     }
26 }
27
28 void show() {
29     for(int i=0; i < len; i++)
30         printf("%d ",*(array + i));
31     printf("\n");
32 }
33
34 void InsertionSort() {
35     int temp;
36     int j;
37     for (int i=0; i < len; i++) {
38         temp = array[i];
39         for (j = i;j>0 && array[j-1] > temp;j--) {
40             array[j] = array[j-1];
41         }
42         array[j] = temp;
43     }
44 }
45
46 int main(int argc, char* argv[]) {
47     // For testing puposes no prefix text will be given for input
48     init();
49
50     InsertionSort();
51     if(argc < 2)
52         show();
53
54     return 0;
55 }
56
```

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

```
1 /**
2  * Data Structures Assignment - 4
3  * CS 392
4  * Problem 4:
5  *      Write a C program to sort the given array element using Merge Sort Algorithm
```

```
6 *
7 * Joydeep Mukherjee
8 * CSE, 3rd Sem, 11000116030
9 * GCETTS 2017
10 *
11 **/
12
13
14
15 #include <stdio.h>
16 #include <stdlib.h>
17
18 int *array;
19 int *temp;
20 int len;
21
22 void Merge(int left, int mid, int right);
23
24 void init() {
25     scanf("%d",&len);
26     array = malloc(sizeof(int)*len);
27     temp = malloc(sizeof(int)*len);
28
29     for (int i=0; i<len; i++) {
30         scanf("%d",(array+i));
31     }
32 }
33
34 void show() {
35     for(int i=0; i < len; i++)
36         printf("%d ",*(array + i));
37     printf("\n");
38 }
39
40 void MergeSort(int left, int right) {
41     int mid;
42     if (right > left) {
43         mid = (right + left) / 2;
44         MergeSort(left, mid);
45         MergeSort(mid + 1, right);
46         Merge(left, mid+1, right);
47     }
48 }
49 void Merge(int left, int mid, int right) {
50     int i, left_end, size, temp_pos;
51
52     left_end = mid - 1;
53     temp_pos = left;
54
55     size = right - left + 1;
56
57     while((left <= left_end) && (mid <= right)) {
58         if(array[left] <= array[mid])
59             temp[temp_pos++] = array[left++];
60         else
61             temp[temp_pos++] = array[mid++];
62     }
63     while( left <= left_end) {
64         temp[temp_pos++] = array[left++];
65     }
66     while( mid <= right) {
```



```
67     temp[temp_pos++] = array[mid++];
68 }
69 for(int i=0; i<= size; i++) {
70     array[right] = temp[right--];
71 }
72 }
73
74 int main(int argc, char* argv[]) {
75     // For testing puposes no prefix text will be given for input
76     init();
77
78     MergeSort(0,len-1);
79     if(argc < 2)
80         show();
81
82     return 0;
83 }
84
85
```

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

```
1 /**
2  * Data Structures Assignment - 4
3  * CS 392
4  * Problem 5:
5  *      Write a C program to sort the given array element using Quick Sort Algorithm
6  *
7  * Joydeep Mukherjee
8  * CSE, 3rd Sem, 11000116030
9  * GCETTS 2017
10 *
11 **/
12
13
14
15
16 #include <stdio.h>
17 #include <stdlib.h>
18
19 int *array;
20 int len;
21
22 void init() {
23     scanf("%d",&len);
24     array = malloc(sizeof(int)*len);
25     for (int i=0; i<len; i++) {
26         scanf("%d",(array+i));
27     }
28 }
29
30 void show() {
31     for(int i=0; i < len; i++)
32         printf("%d ",*(array + i));
33     printf("\n");
34 }
35
36 void QuickSort(int low, int high) {
```

```
37 int pos_pivot = low;
38 int pivot = array[low];
39 int left = low;
40 int right = high;
41 int temp = 0;
42 // Base case for recursion
43 if(high > low) {
44     while (right > left) {
45         while(array[left] <= pivot)
46             left++;
47         while(array[right] > pivot)
48             right--;
49
50         if(left < right) {
51             temp = array[right];
52             array[right] = array[left];
53             array[left] = temp;
54         }
55     }
56
57     array[pos_pivot] = array[right];
58     array[right] = pivot;
59     pos_pivot = right;
60
61
62     QuickSort(low, pos_pivot-1);
63     QuickSort(pos_pivot+1, high);
64 }
65 return;
66 }
67
68 int main(int argc, char* argv[]) {
69     // For testing puposes no prefix text will be given for input
70     init();
71
72     QuickSort(0, len-1);
73     if(argc < 2)
74         show();
75
76     return 0;
77 }
78
79
```

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
```

```
14 #include <stdio.h>
15 #include <stdlib.h>
16
17 int* array;
18 int len;
19
20 void init() {
21     scanf("%d",&len);
22     array = malloc(sizeof(int)*len);
23     for (int i=0; i<len; i++) {
24         scanf("%d",(array+i));
25     }
26 }
27
28 void show() {
29     for(int i=0; i < len; i++)
30         printf("%d ",*(array + i));
31     printf("\n");
32 }
33
34 int getMax(int n) {
35     int mx = array[0];
36     for(int i=0; i < n; i++) {
37         if (array[i] > mx)
38             mx = array[i];
39     }
40     return mx;
41 }
42
43 void countSort(int n, int ex) {
44     int* output = malloc(sizeof(int)*n);
45     int i,count[10] = {0};
46
47     for (i = 0; i < n; i++)
48         count[ (array[i]/ex)%10 ]++;
49
50     for (i = 1; i < n; i++)
51         count[i] += count[i-1];
52
53     for (i = n - 1; i>= 0; i-- ) {
54         output[count[ (array[i]/ex)%10 ] - 1] = array[i];
55         count[ (array[i]/ex)%10 ]--;
56     }
57
58     for (i = 0; i < n; i++)
59         array[i] = output[i];
60 }
61
62 void radixsort(int n) {
63     int m = getMax(n);
64
65     for (int ex = 1;m/ex > 0; ex *= 10)
66         countSort(n,ex);
67 }
68
69 int main(int argc, char* argv[]) {
70     // For testing puposes no prefix text will be given for input
71     init();
72
73     radixsort(len);
74     if(argc < 2)
75         show();
76 }
```

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