

This assignment is to be done individually.

Important Note: The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

DO NOT:

- Give/receive code or proofs to/from other students
- Use search engines to find solutions for the assignment

DO:

- Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
- Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment

Submission Instructions:

- You may type or write your answer as long as it is readable.
 - Submit two files on CourSys
 1. `report.pdf`, which contains a write-up of your solutions to the assignment
 2. `DupRmv.c`, which contains the code you wrote in Problem 2.
-

Problem 1

We have seen in the class lectures that the smallest big-O estimate for a comparison-type sort algorithm is $O(n \log n)$. Are there any more computationally efficient algorithms?

Consider a vector of votes (V) for an array of integer numbers (a). Each element in V declares the number of items in original vector a that are less than or equal to (LE) the corresponding index. For example, for array $a = \{3, 5, 2, 5\}$ the vote vector would be: (16 marks)

$$V = [0, 0, 1, 2, 2, 4]$$

which means there are four numbers LE 5; two numbers LE 4,3; one number LE 2; and no number LE 1,0. Answer the following questions in your write up (no separate code submission).

- Write a pseudo code that generates the voting vector given an array a in linear time $O(n)$. Is the inverse of this operation always unique? (3 marks)
- Given the voting vector, write a pseudo code that reconstructs input vector (a) in linear time $O(n)$ but in sorted order. (Hint: The voting vector values can be used as the final position of the sorted vector ($a_{sorted}[V[j]]$) for the corresponding array element (j)) (3 marks)
- Based on previous parts, complete the following code to sort an input array. Just write a **typed** snippet for the required two blocks in your report (make sure the original code would be functional after adding your two blocks of snippets) (5 marks)

```
1 #include <stdio.h>
2
3 void voting_sort(int a[], int k, int n)
4 {
5     int i, j;
6     int b[15], v[100];
7     /* initialize the voting vector */
8     for (i = 0; i <= k; i++)
9         v[i] = 0;
10    /*Construct the voting vector*/
11    /*
12    Block 1: add your code here
13    */
14
15    /*Reconstruct the initial vector store it in b*/
16    /*
17    Block 2: add your code here
```

```
18  */
19  printf("The Sorted array is : ");
20  for (i = 1; i <= n; i++)
21      printf("%d,", b[i]);
22  }
23  int main()
24  {
25      int n, k = 0, a[15], i;
26      printf("Input number of elements:  ");
27      scanf("%d", &n);
28      printf("Input the array elements one by one: \n");
29      for (i = 1; i <= n; i++)
30      {
31          scanf("%d", &a[i]);
32          if (a[i] > k)
33              k = a[i];
34      }
35      voting_sort(a, k, n);
36      return 0;
37  }
```

- d) What is the total running time of this algorithm? what is the space complexity (required memory)? What are the limitation of this algorithm for sorting (name at least three)? Suggest at least one modification to address any of those issues. (5 marks)

Problem 2

Write a code `DupRmv.c` that receives a list of integer numbers separated by space and terminated by EOF (End of File) as input and output the mean of the non-duplicated elements with 2 decimal points precision. The algorithm should be at most $O(n \log n)$. In your write up, describe the pseudocode and the complexity (14 marks).

IMPORTANT:

DO: use `scanf` to get input and `printf` to show output.

DO NOT:

- print anything other than the float output number, like prompt for input, commenting the output;
- check the credibility of input;
- use `\n` at the end of output;

- include anything other than `<stdio.h>`.

Input sample 1: 2 -2.5 3 2.5 3 EOF

Output: 1.25

Input sample 2: 3 3 3 1 EOF

Output: 2.00

Resources:

- To input EOF in command line, press ctrl+D.
- Reference for `scanf`: <http://www.cplusplus.com/reference/cstdio/scanf/>