

Contents

1	Rubric	6
2	Metadata	7
2.1	Submitted Files	7
2.2	webgrader Runs	8
2.3	diffs	9
3	Written Exercises	10
4	Programming Exercises	18
4.1	csce310h0mework02part01	18
4.1.1	Test 01	18
	diff	18
	Input	18
	Submission Output	18
	Solution Output	18
	stderr	18
4.1.2	Test 02	18
	diff	18
	Input	19
	Submission Output	19
	Solution Output	19
	stderr	19
4.1.3	Test 03	19
	diff	19
	Input	19
	Submission Output	20
	Solution Output	20
	stderr	20
4.1.4	Test 04	20
	diff	20
	Input	20
	Submission Output	20
	Solution Output	21
	stderr	21
4.1.5	Test 05	21
	diff	21
	Input	21
	Submission Output	21
	Solution Output	21
	stderr	22
4.1.6	Test 06	22
	diff	22
	Input	22
	Submission Output	22
	Solution Output	22
	stderr	22
4.1.7	Test 07	22
	diff	22

	Input	23
	Submission Output	23
	Solution Output	23
	stderr	23
4.1.8	Test 08	23
	diff	23
	Input	23
	Submission Output	23
	Solution Output	24
	stderr	24
4.1.9	Test 09	24
	diff	24
	Input	24
	Submission Output	24
	Solution Output	25
	stderr	25
4.1.10	Test 10	25
	diff	25
	Input	25
	Submission Output	25
	Solution Output	26
	stderr	26
4.1.11	Test 11	26
	diff	26
	Input	26
	Submission Output	26
	Solution Output	26
	stderr	27
4.1.12	Test 12	27
	diff	27
	Input	27
	Submission Output	27
	Solution Output	27
	stderr	28
4.1.13	Test 13	28
	diff	28
	Input	28
	Submission Output	28
	Solution Output	28
	stderr	29
4.1.14	Test 14	29
	diff	29
	Input	29
	Submission Output	29
	Solution Output	29
	stderr	29
4.1.15	Test 15	30
	diff	30
	Input	30
	Submission Output	30
	Solution Output	30
	stderr	30
4.1.16	Source Code	30
4.2	csce310h0mework02part02	31
4.2.1	Test 01	31
	diff	31
	Input	31
	Submission Output	31
	Solution Output	31
	stderr	31

4.2.2	Test 02	32
	diff	32
	Input	32
	Submission Output	32
	Solution Output	32
	stderr	32
4.2.3	Test 03	32
	diff	32
	Input	32
	Submission Output	32
	Solution Output	32
	stderr	32
4.2.4	Test 04	32
	diff	32
	Input	32
	Submission Output	33
	Solution Output	33
	stderr	33
4.2.5	Test 05	33
	diff	33
	Input	33
	Submission Output	33
	Solution Output	33
	stderr	33
4.2.6	Test 06	33
	diff	33
	Input	33
	Submission Output	33
	Solution Output	33
	stderr	33
4.2.7	Test 07	34
	diff	34
	Input	34
	Submission Output	34
	Solution Output	34
	stderr	34
4.2.8	Test 08	34
	diff	34
	Input	34
	Submission Output	34
	Solution Output	34
	stderr	34
4.2.9	Test 09	34
	diff	34
	Input	34
	Submission Output	35
	Solution Output	35
	stderr	35
4.2.10	Test 10	35
	diff	35
	Input	35
	Submission Output	35
	Solution Output	35
	stderr	35
4.2.11	Test 11	35
	diff	35
	Input	35
	Submission Output	35
	Solution Output	35
	stderr	35

4.2.12	Test 12	36
	diff	36
	Input	36
	Submission Output	36
	Solution Output	36
	stderr	36
4.2.13	Test 13	36
	diff	36
	Input	36
	Submission Output	36
	Solution Output	36
	stderr	36
4.2.14	Test 14	36
	diff	36
	Input	36
	Submission Output	37
	Solution Output	37
	stderr	37
4.2.15	Test 15	37
	diff	37
	Input	37
	Submission Output	37
	Solution Output	37
	stderr	37
4.2.16	Source Code	37
4.3	csce310h0mework02part03	38
4.3.1	Test 01	38
	diff	38
	Input	38
	Submission Output	38
	Solution Output	38
	stderr	38
4.3.2	Test 02	38
	diff	38
	Input	38
	Submission Output	39
	Solution Output	39
	stderr	39
4.3.3	Test 03	39
	diff	39
	Input	39
	Submission Output	39
	Solution Output	39
	stderr	39
4.3.4	Test 04	39
	diff	39
	Input	39
	Submission Output	39
	Solution Output	39
	stderr	40
4.3.5	Test 05	40
	diff	40
	Input	40
	Submission Output	40
	Solution Output	40
	stderr	40
4.3.6	Test 06	40
	diff	40
	Input	40
	Submission Output	40

	Solution Output	40
	stderr	40
4.3.7	Test 07	40
	diff	40
	Input	41
	Submission Output	41
	Solution Output	41
	stderr	41
4.3.8	Test 08	41
	diff	41
	Input	41
	Submission Output	41
	Solution Output	41
	stderr	41
4.3.9	Test 09	41
	diff	41
	Input	41
	Submission Output	41
	Solution Output	42
	stderr	42
4.3.10	Test 10	42
	diff	42
	Input	42
	Submission Output	42
	Solution Output	42
	stderr	42
4.3.11	Source Code	42

Chapter 1

Rubric

Question	Points
Question 1	10
Question 2	10
Question 3	10
Question 4	10
Question 5	10
Question 6	10
placeNumbers	
Test Cases	1 × 15
Compilation	5
placeNumbers Total	20
closestNumbers	
Test Cases	1 × 15
Compilation	5
closestNumbers Total	20
Total	100

Chapter 2

Metadata

2.1 Submitted Files

handin.time					
1	09/26/2019	13:34:14	fsandhu:	csce310h0mework02part02.cpp	- OK
2	09/26/2019	13:34:17	fsandhu:	csce310h0mework02part02.h	- OK
3	09/26/2019	13:35:33	fsandhu:	csce310h0mework02part02.cpp	- OK
4	09/26/2019	13:42:55	fsandhu:	csce310h0mework02part02.cpp	- OK
5	09/26/2019	13:45:42	fsandhu:	csce310h0mework02part02.cpp	- OK
6	09/26/2019	13:47:21	fsandhu:	csce310h0mework02part02.cpp	- OK
7	09/26/2019	13:48:52	fsandhu:	csce310h0mework02part02.cpp	- OK
8	09/26/2019	14:40:03	fsandhu:	csce310h0mework02part01.cpp	- OK
9	09/26/2019	14:40:06	fsandhu:	csce310h0mework02part01.h	- OK
10	09/26/2019	14:44:03	fsandhu:	csce310h0mework02part01.cpp	- OK
11	09/26/2019	14:53:05	fsandhu:	csce310h0mework02part01.cpp	- OK
12	09/26/2019	14:54:33	fsandhu:	csce310h0mework02part01.cpp	- OK
13	09/26/2019	14:56:20	fsandhu:	csce310h0mework02part02.cpp	- OK
14	09/26/2019	14:57:56	fsandhu:	csce310h0mework02part02.cpp	- OK
15	09/26/2019	15:08:10	fsandhu:	csce310h0mework02part01.cpp	- OK
16	09/26/2019	15:15:32	fsandhu:	csce310h0mework02part01.cpp	- OK
17	09/26/2019	20:25:13	fsandhu:	csce310h0mework02part01.cpp	- OK
18	09/26/2019	20:27:30	fsandhu:	csce310h0mework02part01.cpp	- OK
19	09/26/2019	20:33:47	fsandhu:	csce310h0mework02part01.cpp	- OK
20	09/26/2019	20:33:50	fsandhu:	csce310h0mework02part01.h	- OK
21	09/26/2019	20:33:52	fsandhu:	csce310h0mework02part02.cpp	- OK
22	09/26/2019	20:33:54	fsandhu:	csce310h0mework02part02.h	- OK
23	09/26/2019	21:04:20	fsandhu:	csce310h0mework02part03.cpp	- OK
24	09/26/2019	21:04:22	fsandhu:	csce310h0mework02part03.h	- OK
25	09/26/2019	21:06:24	fsandhu:	csce310h0mework02part03.cpp	- OK
26	09/26/2019	21:06:52	fsandhu:	csce310h0mework02part03.cpp	- OK
27	09/27/2019	10:44:21	fsandhu:	csce310h0mework02part03.cpp	- OK
28	09/28/2019	01:12:05	fsandhu:	csce310h0mework02part03.cpp	- OK
29	09/28/2019	01:12:59	fsandhu:	csce310h0mework02part03.cpp	- OK
30	09/28/2019	01:14:17	fsandhu:	csce310h0mework02part03.cpp	- OK
31	09/28/2019	01:30:02	fsandhu:	csce310h0mework02part03.cpp	- OK
32	09/28/2019	01:32:11	fsandhu:	csce310h0mework02part03.cpp	- OK
33	09/28/2019	01:32:20	fsandhu:	csce310h0mework02part03.h	- OK
34	09/28/2019	01:33:42	fsandhu:	csce310h0mework02part03.cpp	- OK
35	09/28/2019	01:33:47	fsandhu:	csce310h0mework02part03.h	- OK
36	09/28/2019	01:45:47	fsandhu:	csce310h0mework02part03.cpp	- OK
37	09/28/2019	01:47:27	fsandhu:	csce310h0mework02part03.cpp	- OK
38	09/28/2019	01:51:22	fsandhu:	csce310h0mework02part03.cpp	- OK
39	09/28/2019	01:51:25	fsandhu:	csce310h0mework02part03.h	- OK
40	09/28/2019	01:52:40	fsandhu:	csce310h0mework02part03.cpp	- OK
41	09/28/2019	01:56:38	fsandhu:	csce310h0mework02part03.cpp	- OK

42	09/28/2019	01:56:40	fsandhu:	csce310h0mework02part03.h	- OK	
43	09/28/2019	01:58:17	fsandhu:	csce310h0mework02part03.cpp		- OK
44	09/28/2019	01:58:19	fsandhu:	csce310h0mework02part03.h	- OK	
45	09/28/2019	02:03:55	fsandhu:	csce310h0mework02part03.cpp		- OK
46	10/01/2019	14:16:07	fsandhu:	csce310h0mework02part03.cpp		- OK
47	10/01/2019	14:16:50	fsandhu:	csce310h0mework02part03.cpp		- OK
48	10/09/2019	18:33:38	fsandhu:	csce310h0mework02part03.cpp		- OK
49	10/09/2019	18:34:26	fsandhu:	csce310h0mework02part03.cpp		- OK
50	10/09/2019	21:25:25	fsandhu:	csce310hw02.pdf	- OK	

2.2 webgrader Runs

				webgrader.time
1	2019-09-26T13:34:23-0500	10.43.1.171	fsandhu	002
2	2019-09-26T13:35:36-0500	10.43.1.171	fsandhu	002
3	2019-09-26T13:43:13-0500	10.43.1.171	fsandhu	002
4	2019-09-26T13:45:44-0500	10.43.1.171	fsandhu	002
5	2019-09-26T13:47:18-0500	10.43.1.171	fsandhu	002
6	2019-09-26T13:47:24-0500	10.43.1.171	fsandhu	002
7	2019-09-26T14:40:16-0500	10.43.1.171	fsandhu	002
8	2019-09-26T14:44:07-0500	10.43.1.171	fsandhu	002
9	2019-09-26T14:53:07-0500	10.43.1.171	fsandhu	002
10	2019-09-26T14:54:37-0500	10.43.1.171	fsandhu	002
11	2019-09-26T14:56:24-0500	10.43.1.171	fsandhu	002
12	2019-09-26T14:57:58-0500	10.43.1.171	fsandhu	002
13	2019-09-26T15:08:12-0500	10.43.1.171	fsandhu	002
14	2019-09-26T15:15:34-0500	10.43.1.171	fsandhu	002
15	2019-09-26T20:25:20-0500	76.84.50.181	fsandhu	002
16	2019-09-26T20:27:33-0500	76.84.50.181	fsandhu	002
17	2019-09-26T20:33:39-0500	76.84.50.181	fsandhu	002
18	2019-09-26T20:33:56-0500	76.84.50.181	fsandhu	002
19	2019-09-26T21:04:25-0500	76.84.50.181	fsandhu	002
20	2019-09-26T21:06:27-0500	76.84.50.181	fsandhu	002
21	2019-09-26T21:06:59-0500	76.84.50.181	fsandhu	002
22	2019-09-27T10:44:39-0500	10.43.118.108	fsandhu	002
23	2019-09-28T01:12:11-0500	97.98.163.171	fsandhu	002
24	2019-09-28T01:13:02-0500	97.98.163.171	fsandhu	002
25	2019-09-28T01:14:19-0500	97.98.163.171	fsandhu	002
26	2019-09-28T01:30:05-0500	97.98.163.171	fsandhu	002
27	2019-09-28T01:32:23-0500	97.98.163.171	fsandhu	002
28	2019-09-28T01:33:50-0500	97.98.163.171	fsandhu	002
29	2019-09-28T01:45:49-0500	97.98.163.171	fsandhu	002
30	2019-09-28T01:47:29-0500	97.98.163.171	fsandhu	002
31	2019-09-28T01:51:27-0500	97.98.163.171	fsandhu	002
32	2019-09-28T01:52:42-0500	97.98.163.171	fsandhu	002
33	2019-09-28T01:56:42-0500	97.98.163.171	fsandhu	002
34	2019-09-28T01:58:07-0500	97.98.163.171	fsandhu	002
35	2019-09-28T01:58:21-0500	97.98.163.171	fsandhu	002
36	2019-09-28T02:04:07-0500	97.98.163.171	fsandhu	002
37	2019-09-30T13:50:40-0500	10.43.63.145	fsandhu	002
38	2019-10-01T14:16:19-0500	10.43.114.9	fsandhu	002
39	2019-10-01T14:16:53-0500	10.43.114.9	fsandhu	002
40	2019-10-01T14:33:44-0500	10.43.114.9	fsandhu	002
41	2019-10-08T14:50:16-0500	10.43.23.54	fsandhu	002
42	2019-10-08T14:51:02-0500	10.43.23.54	fsandhu	002
43	2019-10-08T14:51:54-0500	10.43.23.54	fsandhu	002
44	2019-10-08T14:59:55-0500	10.43.23.54	fsandhu	002
45	2019-10-08T15:00:17-0500	10.43.23.54	fsandhu	002
46	2019-10-08T15:00:30-0500	10.43.23.54	fsandhu	002

47	2019-10-08T15:00:45-0500	10.43.23.54	fsandhu	002
48	2019-10-08T15:01:09-0500	10.43.23.54	fsandhu	002
49	2019-10-08T15:02:46-0500	10.43.23.54	fsandhu	002
50	2019-10-08T15:03:00-0500	10.43.23.54	fsandhu	002
51	2019-10-08T15:03:22-0500	10.43.23.54	fsandhu	002
52	2019-10-08T15:04:08-0500	10.43.23.54	fsandhu	002
53	2019-10-08T15:04:21-0500	10.43.23.54	fsandhu	002
54	2019-10-09T14:28:00-0500	10.43.67.76	fsandhu	002
55	2019-10-09T14:28:54-0500	10.43.67.76	fsandhu	002
56	2019-10-09T18:33:45-0500	76.84.50.181	fsandhu	002
57	2019-10-09T18:34:28-0500	76.84.50.181	fsandhu	002
58	2019-10-09T18:34:55-0500	76.84.50.181	fsandhu	002
59	2019-10-09T20:33:02-0500	76.84.50.181	fsandhu	002
60	2019-10-09T21:25:44-0500	76.84.50.181	fsandhu	002
61	2019-10-11T10:37:53-0500	10.43.47.170	fsandhu	002
62	2019-12-15T19:52:19-0600	76.84.219.87	fsandhu	002

2.3 diffs

submission.diffs

Chapter 3

Written Exercises

Q1) The algorithm is a recursive algorithm but it is incorrect because it does not have correct base cases. It does not take into account the last leaf and returns 0 if the node passed in is empty. The algorithm will return 0 if we run it as of now.

Corrected algorithm:

LeafCounter.

Input: A binary tree T

Output: The number of leaves in T

if $T = \emptyset$

return 0

else if $T_{\text{right}} = \emptyset$ and $T_{\text{left}} = \emptyset$

return 1

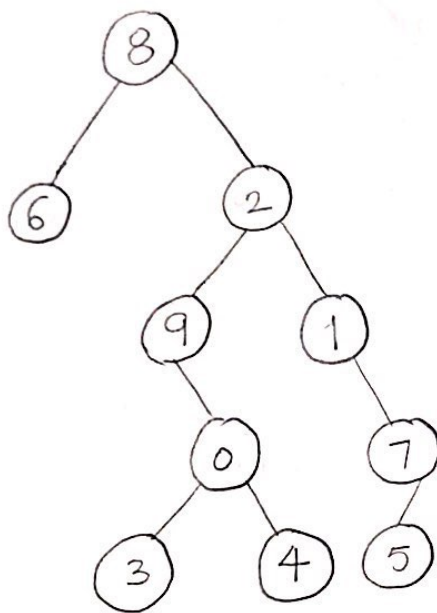
else

return LeafCounter(T_{left}) + LeafCounter(T_{right})

Q2).

a) PRE : 8 6 2 9 0 3 4 1 7 5 NLR

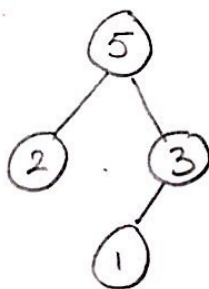
IN : 6 8 9 3 0 4 2 1 5 7 LNR



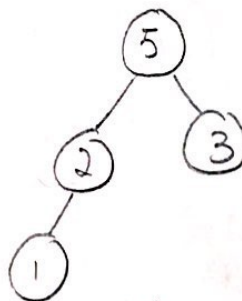
Resulting binary tree

b) PRE : 5 2 3 1

IN : 1 2 5 3



PRE



IN

They are traversals of the same binary tree

c) Construct Tree

// recursive algorithm

Input: preorder and inorder traversal of a tree

$[i_0, i_1, \dots, i_{n-1}]$

$[p_0, p_1, \dots, p_{n-1}]$

Output: Binary Tree OR invalid input

for int $i \rightarrow n-1$.

if $\text{inorder}[i] = \text{preorder}[0]$

print $\text{inorder}[i]$.

break

else

return error [invalid input]

return Construct Tree ($\text{inorder}[0, 1, \dots, i_{k-1}]$
 $\text{preorder}[p_1, p_2, \dots, p_k]$)

Construct Tree ($\text{inorder}[i_{k+1}, i_{k+2}, \dots, i_{n-1}]$
 $\text{preorder}[p_{k+1}, p_{k+2}, \dots, p_{n-1}]$)

// recursively build left and right subtrees.

Q3). If the search is only done once, then

linear search would be a better option

because it has only n comparisons. However, if we are doing multiple searches, presorting the algorithm would be better because we can now use binary search which requires $\log_2(n)$ comparison. for justification, we can calculate after how many searches will presort be better.

$$n \log_2(n) + K \log_2(n) \leq K n \quad \text{where } K \text{ is the number of searches done.}$$

$$n \log_2(n) \leq K (n - \log_2(n))$$

$$\frac{n \log_2(n)}{n - \log_2(n)} \leq K$$

for $n = 10^4$ elements

$$K \geq \frac{10^4 \log_2(10^4)}{10^4 - \log_2(10^4)} \Rightarrow K \geq 13.305392202$$

$$\Rightarrow K \approx 14$$

after 14 searches, presort will be justified for 10^4 elements

for $n = 10^7$ elements

$$K \geq \frac{10^7 \log_2(10^7)}{10^7 - \log_2(10^7)} \Rightarrow K \geq 23.2535507368$$

$$K \approx 24$$

after 24 searches, presort will be justified for 10^7 elements

Q4)

a) Input : An array of n numbers

Output : closest distance b/w two numbers in array

Sort the array with the most efficient sorting algorithm based on size.
we use merge sort as an example.

$$\text{closestDistance} = |a[0] - a[1]|$$

for $i = 1 \rightarrow \text{size} - 2$.

if $(\text{closestDistance} > |a[i] - a[i+1]|)$

set $\text{closestDistance} = |a[i] - a[i+1]|$

return closestDistance.

$$\text{time complexity : } \underbrace{n \log_2(n)}_{\text{mergesort worst case}} + \underbrace{n-1}_{\text{closestDist worst case}}$$

$$\Rightarrow O(n \log_2(n))$$

b). Brute force worst case time complexity : $O(n^2)$

Presort time complexity $\Rightarrow O(n \log_2(n))$.
worst case

Presort is way efficient than brute force.

Q5) programming part 1. refer for algorithm with comments.

Input : a list of n distinct integers
a sequence of boxes with signs
preset

Output : place numbers into boxes accordingly.

(mergesort) sort the list in ascending order.

numbersPlaced from end = 0

for $i = 0 \rightarrow \text{sizeOfList}$

| if $\text{sign}[i] = ">"$

place into next available box ($\text{list}[\text{sizeOfList} - \text{numbersPlaced from End}]$)
| numberPlaced from End ++

else if $\text{sign}[i] = "<"$

place into next available box ($\text{list}[i - \text{numbersPlaced from End}]$)

for last box \rightarrow

place into box ($\text{list}[\text{sizeOfList} - \text{numbersPlaced from End}]$)

return sequence of boxes.

time complexity $\underbrace{n \log_2(n)}_{\text{merge sort}} + n$

$\Rightarrow O(n \log_2(n))$.

Q6)

number of additions = $(n+1)$.

number of multiplications = $(n+1) + (1+2+3+\dots+n)$
 $= (n+1) + \frac{n(n+1)}{2}$

$$= \frac{2(n+1) + n(n+1)}{2}$$

$$= \frac{(n+2)(n+1)}{2}$$

$$= \frac{n^2 + 2n + n + 2}{2}$$

$$= \frac{n^2 + 3n + 2}{2}$$

Chapter 4

Programming Exercises

4.1 csce310h0mework02part01

4.1.1 Test 01

diff

part01test01.diff

Input

part01test01.input

<><>><

10 16 26 51 60 74 96

Submission Output

part01test01.output

10 < 96 > 16 < 74 > 60 > 26 < 51

VALID

Solution Output

part01test01.solution

10

^

96

v

16

^

74

v

60

v

26

^

51

VALID

stderr

part01test01.err

4.1.2 Test 02

diff

part01test02.diff

Input

part01test02.input

```
>><><><><><><<
4 7 9 10 15 45 51 60 61 63 64 76 80 81 91
```

Submission Output

part01test02.output

```
91 > 81 > 4 < 80 > 7 < 76 > 9 < 10 < 64 > 15 < 45 < 63 > 51 < 60 < 61
```

VALID

Solution Output

part01test02.solution

```
91
v
81
v
4
^
80
v
7
^
76
v
9
^
10
^
64
v
15
^
45
^
63
v
51
^
60
^
61
VALID
stderr
```

part01test02.err

4.1.3 Test 03

diff

part01test03.diff

Input

part01test03.input

<<><><>><>

2 19 28 30 43 50 52 59 84 92 93 94

Submission Output

part01test03.output

2 < 19 < 94 > 28 < 93 > 30 < 92 > 84 > 59 > 43 < 52 > 50

VALID

Solution Output

part01test03.solution

2

^

19

^

94

v

28

^

93

v

30

^

92

v

84

v

59

v

43

^

52

v

50

VALID

stderr

part01test03.err

4.1.4 Test 04

diff

part01test04.diff

Input

part01test04.input

<<><><<<<>>>

4 21 24 25 28 29 33 37 40 43 84 90 98

Submission Output

part01test04.output

4 < 21 < 98 > 24 < 90 > 25 < 28 < 29 < 33 < 84 > 43 > 40 > 37

VALID

Solution Output

part01test04.solution

```
4
~
21
~
98
v
24
~
90
v
25
~
28
~
29
~
33
~
84
v
43
v
40
v
37
VALID
stderr
```

part01test04.err

4.1.5 Test 05

diff

part01test05.diff

Input

part01test05.input

```
<<><<><>><
4 16 27 30 48 51 86 96 97 98 99
```

Submission Output

part01test05.output

```
4 < 16 < 99 > 27 < 30 < 98 > 48 < 97 > 96 > 51 < 86
```

VALID

Solution Output

part01test05.solution

```
4
~
16
~
```

```
99
v
27
^
30
^
98
v
48
^
97
v
96
v
51
^
86
VALID
stderr
```

part01test05.err

4.1.6 Test 06

diff

part01test06.diff

Input

part01test06.input

```
<><
26 72 90 95
```

Submission Output

part01test06.output

```
26 < 95 > 72 < 90
```

VALID

Solution Output

part01test06.solution

```
26
^
95
v
72
^
90
VALID
stderr
```

part01test06.err

4.1.7 Test 07

diff

part01test07.diff

Input

part01test07.input

```
><>><<><><>
17 40 42 51 58 61 74 76 77 92 96 100
```

Submission Output

part01test07.output

```
100 > 17 < 96 > 92 > 40 < 42 < 77 > 51 < 76 > 58 < 74 > 61
```

VALID

Solution Output

part01test07.solution

```
100
v
17
^
96
v
92
v
40
^
42
^
77
v
51
^
76
v
58
^
74
v
61
VALID
stderr
```

part01test07.err

4.1.8 Test 08

diff

part01test08.diff

Input

part01test08.input

```
><<<><>><><<>><>
12 23 28 29 30 31 35 38 54 56 60 70 73 75 81 87 88
```

Submission Output

part01test08.output

88 > 12 < 23 < 28 < 87 > 29 < 81 > 75 > 30 < 73 > 31 < 35 < 70 > 60 > 38 < 56 > 54

VALID

Solution Output

part01test08.solution

88

v

12

^

23

^

28

^

87

v

29

^

81

v

75

v

30

^

73

v

31

^

35

^

70

v

60

v

38

^

56

v

54

VALID

stderr

part01test08.err

4.1.9 Test 09

diff

part01test09.diff

Input

part01test09.input

><>><>><><><><>

6 13 24 26 35 47 48 60 66 70 74 81 85 90 95 98

Submission Output

part01test09.output

98 > 6 < 95 > 90 > 13 < 85 > 81 > 24 < 74 > 26 < 70 > 35 < 66 > 47 < 60 > 48

VALID

Solution Output

part01test09.solution

98

v

6

^

95

v

90

v

13

^

85

v

81

v

24

^

74

v

26

^

70

v

35

^

66

v

47

^

60

v

48

VALID

stderr

part01test09.err

4.1.10 Test 10

diff

part01test10.diff

Input

part01test10.input

>><<>><<><>

3 8 9 18 40 49 50 56 57 66 73 78

Submission Output

part01test10.output

78 > 73 > 3 < 8 < 66 > 57 > 56 > 9 < 50 > 18 < 49 > 40

VALID

Solution Output

part01test10.solution

78
v
73
v
3
^
8
^
66
v
57
v
56
v
9
^
50
v
18
^
49
v
40
VALID
stderr

part01test10.err

4.1.11 Test 11

diff

part01test11.diff

Input

part01test11.input

<>><>><
4 5 27 45 47 70 90 97

Submission Output

part01test11.output

4 < 97 > 90 > 5 < 70 > 47 > 27 < 45

VALID

Solution Output

part01test11.solution

4

```

^
97
v
90
v
5
^
70
v
47
v
27
^
45
VALID
stderr

```

part01test11.err

4.1.12 Test 12

diff

part01test12.diff

Input

part01test12.input

```

><>>><><><><><><><><
3 7 9 13 17 28 29 33 34 42 47 49 56 57 58 62 79 83 87 95

```

Submission Output

part01test12.output

```

95 > 3 < 87 > 83 > 79 > 7 < 62 > 9 < 58 > 13 < 57 > 17 < 28 < 56 > 49 > 29 < 47 > 42 >
    33 < 34

```

VALID

Solution Output

part01test12.solution

```

95
v
3
^
87
v
83
v
79
v
7
^
62
v
9
^
58

```

```
v
13
^
57
v
17
^
28
^
56
v
49
v
29
^
47
v
42
v
33
^
34
VALID
stderr
```

part01test12.err

4.1.13 Test 13

diff

part01test13.diff

Input

part01test13.input

```
<>><
10 33 53 62 91
```

Submission Output

part01test13.output

```
10 < 91 > 62 > 33 < 53
```

VALID

Solution Output

part01test13.solution

```
10
^
91
v
62
v
33
^
53
VALID
```

stderr

part01test13.err

4.1.14 Test 14

diff

part01test14.diff

Input

part01test14.input

```
<><><><><><><<
18 20 30 31 38 44 45 48 65 77 79 88 91 98 100
```

Submission Output

part01test14.output

```
18 < 100 > 20 < 98 > 30 < 91 > 31 < 88 > 38 < 44 < 79 > 77 > 45 < 48 < 65
```

VALID

Solution Output

part01test14.solution

```
18
^
100
v
20
^
98
v
30
^
91
v
31
^
88
v
38
^
44
^
79
v
77
v
45
^
48
^
65
VALID
stderr
```

part01test14.err

4.1.15 Test 15

diff

part01test15.diff

Input

part01test15.input

<>><>><

14 22 27 37 39 43 69 74 88

Submission Output

part01test15.output

14 < 88 > 74 > 22 < 69 > 43 > 39 > 27 < 37

VALID

Solution Output

part01test15.solution

14

~

88

v

74

v

22

~

69

v

43

v

39

v

27

~

37

VALID

stderr

part01test15.err

4.1.16 Source Code

csce310h0mework02part01.h

```
1 #ifndef CSCE310HOMEWORK02PART01_H
2 #define CSCE310HOMEWORK02PART01_H
3
4 #include <string>
5 #include <vector>
6
7 using namespace std;
8
9 vector<int> placeNumbers( vector<int> , string );
10
11 #endif
```

```

1  /*
2  * Author: Fateh Karan Singh Sandhu
3  * NUID: 17286643
4  *
5  * This function takes in a vector and string of signs and outputs a new vector
6  * with all values placed in accordance with the signs
7  */
8
9  #include "csce310h0mework02part01.h"
10 #include <string>
11 #include <vector>
12 #include <algorithm>
13
14 using namespace std;
15
16 vector<int> placeNumbers( vector<int> numbers , string signs ){
17
18     int numbersPlacedFromEnd = 0; //count for numbers placed from the end of the
    vector
19     vector<int> numbersPlaced; //new vector with numbers placed in accordance to signs
20     for (int i = 0 ; i < numbers.size() ; i++) {
21         if (signs[i] == '>') {
22             numbersPlaced.push_back(numbers[numbers.size()-1-numbersPlacedFromEnd]);
23             //add to new vector
24             numbersPlacedFromEnd++; //iterate variable for every number added from end
25         } else if (signs[i] == '<') {
26             numbersPlaced.push_back(numbers[i-numbersPlacedFromEnd]);
27         }
28     }
29     numbersPlaced.push_back(numbers[numbers.size()-1-numbersPlacedFromEnd]); //insert
    last number into the vector
30     return numbersPlaced; //return new vector
31 }

```

4.2 csce310h0mework02part02

4.2.1 Test 01

diff

part02test01.diff

Input

part02test01.input

6349 5080 3861 2182 -61 -1070 -3151

Submission Output

part02test01.output

The closest numbers are 1009.000000 apart.

Solution Output

part02test01.solution

The closest numbers are 1009.000000 apart.

stderr

part02test01.err

4.2.2 Test 02

diff

part02test02.diff

Input

part02test02.input

8397 -4088 -6723

Submission Output

part02test02.output

The closest numbers are 2635.000000 apart.

Solution Output

part02test02.solution

The closest numbers are 2635.000000 apart.

stderr

part02test02.err

4.2.3 Test 03

diff

part02test03.diff

Input

part02test03.input

8506 8502 8076 7953 5421 5035 2421 224 -3749 -6154

Submission Output

part02test03.output

The closest numbers are 4.000000 apart.

Solution Output

part02test03.solution

The closest numbers are 4.000000 apart.

stderr

part02test03.err

4.2.4 Test 04

diff

part02test04.diff

Input

part02test04.input

9231 1136 390 208 -960 -975 -2258 -5332 -6129 -9195

Submission Output

part02test04.output

The closest numbers are 15.000000 apart.

Solution Output

part02test04.solution

The closest numbers are 15.000000 apart.

stderr

part02test04.err

4.2.5 Test 05

diff

part02test05.diff

Input

part02test05.input

9712 8980 8877 4992 3827 1743 -3781 -6009 -8375 -9441

Submission Output

part02test05.output

The closest numbers are 103.000000 apart.

Solution Output

part02test05.solution

The closest numbers are 103.000000 apart.

stderr

part02test05.err

4.2.6 Test 06

diff

part02test06.diff

Input

part02test06.input

6020 5552 2242 -22 -4703 -5817 -9722

Submission Output

part02test06.output

The closest numbers are 468.000000 apart.

Solution Output

part02test06.solution

The closest numbers are 468.000000 apart.

stderr

part02test06.err

4.2.7 Test 07

diff

part02test07.diff

Input

part02test07.input

6970 5861 2579 1471 1457 -392 -7644 -8043 -8517 -9296

Submission Output

part02test07.output

The closest numbers are 14.000000 apart.

Solution Output

part02test07.solution

The closest numbers are 14.000000 apart.

stderr

part02test07.err

4.2.8 Test 08

diff

part02test08.diff

Input

part02test08.input

1418 400 -9182

Submission Output

part02test08.output

The closest numbers are 1018.000000 apart.

Solution Output

part02test08.solution

The closest numbers are 1018.000000 apart.

stderr

part02test08.err

4.2.9 Test 09

diff

part02test09.diff

Input

part02test09.input

7252 6645 5741 5203 1904 -762 -1482 -6490 -9525

Submission Output

part02test09.output

The closest numbers are 538.000000 apart.

Solution Output

part02test09.solution

The closest numbers are 538.000000 apart.

stderr

part02test09.err

4.2.10 Test 10

diff

part02test10.diff

Input

part02test10.input

6168 -5176 -8589

Submission Output

part02test10.output

The closest numbers are 3413.000000 apart.

Solution Output

part02test10.solution

The closest numbers are 3413.000000 apart.

stderr

part02test10.err

4.2.11 Test 11

diff

part02test11.diff

Input

part02test11.input

9578 599 -4904

Submission Output

part02test11.output

The closest numbers are 5503.000000 apart.

Solution Output

part02test11.solution

The closest numbers are 5503.000000 apart.

stderr

part02test11.err

4.2.12 Test 12

diff

part02test12.diff

Input

part02test12.input

6757 5818

Submission Output

part02test12.output

The closest numbers are 939.000000 apart.

Solution Output

part02test12.solution

The closest numbers are 939.000000 apart.

stderr

part02test12.err

4.2.13 Test 13

diff

part02test13.diff

Input

part02test13.input

8665 7053 6059 -270 -8186 -8653

Submission Output

part02test13.output

The closest numbers are 467.000000 apart.

Solution Output

part02test13.solution

The closest numbers are 467.000000 apart.

stderr

part02test13.err

4.2.14 Test 14

diff

part02test14.diff

Input

part02test14.input

5991 5825 4966 2230 1482 173 -3642

Submission Output

part02test14.output

The closest numbers are 166.000000 apart.

Solution Output

part02test14.solution

The closest numbers are 166.000000 apart.

stderr

part02test14.err

4.2.15 Test 15

diff

part02test15.diff

Input

part02test15.input

8221 -2097 -4044 -4791

Submission Output

part02test15.output

The closest numbers are 747.000000 apart.

Solution Output

part02test15.solution

The closest numbers are 747.000000 apart.

stderr

part02test15.err

4.2.16 Source Code

csce310h0mework02part02.h

```
1 #ifndef CSCE310HOMEWORK02PART02_H
2 #define CSCE310HOMEWORK02PART02_H
3
4 #include <vector>
5
6 using namespace std;
7
8 double closestNumbers( vector<double> );
9
10 #endif
```

csce310h0mework02part02.cpp

```
1 /*
2  * Author: Fateh Karan Singh Sandhu
3  * NUID: 17286643
4  *
5  * This function takes in a vector of multiple values and returns the
6  * closest difference between two values.
```

```

7  */
8
9  #include "csce310h0mework02part02.h"
10 #include <vector>
11 #include <iostream>
12 #include <cmath>
13
14     using namespace std;
15
16     double closestNumbers( vector<double> numbers )
17     {
18         double closestDistance = abs(numbers[0]-numbers[1]);
19         if (numbers.size() == 2) {
20             return closestDistance; // if vector is of size 2, return the difference
of first 2 and exit
21         } else {
22             for (int i = 1 ; i < numbers.size()-1 ; i++) {
23                 if (abs(numbers[i] - numbers[i+1]) < closestDistance) {
24                     closestDistance = abs(numbers[i] - numbers[i+1]); // update new
closest distance
25                 }
26             }
27             return closestDistance; //return closest distance
28         }
29     }

```

4.3 csce310h0mework02part03

4.3.1 Test 01

diff

part03test01.diff

Input

part03test01.input

11 45 76 77 84 97

Submission Output

part03test01.output

On average, 2.333333 comparisons are needed.

Solution Output

part03test01.solution

On average, 2.333333 comparisons are needed.

stderr

part03test01.err

4.3.2 Test 02

diff

part03test02.diff

Input

part03test02.input

18 19 25 29 33 35 36 39 47 50 51 52 53 70 81 82

Submission Output

part03test02.output

On average, 3.375000 comparisons are needed.

Solution Output

part03test02.solution

On average, 3.375000 comparisons are needed.

stderr

part03test02.err

4.3.3 Test 03

diff

part03test03.diff

Input

part03test03.input

20 48 57 63

Submission Output

part03test03.output

On average, 2.000000 comparisons are needed.

Solution Output

part03test03.solution

On average, 2.000000 comparisons are needed.

stderr

part03test03.err

4.3.4 Test 04

diff

part03test04.diff

Input

part03test04.input

23 24 42 45 55 59 61 62 66

Submission Output

part03test04.output

On average, 2.777778 comparisons are needed.

Solution Output

part03test04.solution

On average, 2.777778 comparisons are needed.

stderr

part03test04.err

4.3.5 Test 05

diff

part03test05.diff

Input

part03test05.input

11 19 32 36 40 45 55 63 72 87 96 97

Submission Output

part03test05.output

On average, 3.083333 comparisons are needed.

Solution Output

part03test05.solution

On average, 3.083333 comparisons are needed.

stderr

part03test05.err

4.3.6 Test 06

diff

part03test06.diff

Input

part03test06.input

24 25 36 41 42 48 56 60 70 76 87 88 98

Submission Output

part03test06.output

On average, 3.153846 comparisons are needed.

Solution Output

part03test06.solution

On average, 3.153846 comparisons are needed.

stderr

part03test06.err

4.3.7 Test 07

diff

part03test07.diff

Input

part03test07.input

13 15 21 34 41 48 49 57 58 63 67 74 84 88 90 92

Submission Output

part03test07.output

On average, 3.375000 comparisons are needed.

Solution Output

part03test07.solution

On average, 3.375000 comparisons are needed.

stderr

part03test07.err

4.3.8 Test 08

diff

part03test08.diff

Input

part03test08.input

38 54 61 62 69 77 93

Submission Output

part03test08.output

On average, 2.428571 comparisons are needed.

Solution Output

part03test08.solution

On average, 2.428571 comparisons are needed.

stderr

part03test08.err

4.3.9 Test 09

diff

part03test09.diff

Input

part03test09.input

15 17 18 19 21 33 39 48 49 50 54 57 65 73 80 89 92

Submission Output

part03test09.output

On average, 3.470588 comparisons are needed.

Solution Output

part03test09.solution

On average, 3.470588 comparisons are needed.

stderr

part03test09.err

4.3.10 Test 10

diff

part03test10.diff

Input

part03test10.input

22 41 49 50 52 76

Submission Output

part03test10.output

On average, 2.333333 comparisons are needed.

Solution Output

part03test10.solution

On average, 2.333333 comparisons are needed.

stderr

part03test10.err

4.3.11 Source Code

csce310h0mework02part03.h

```
1 #ifndef CSCE310HOMEWORK02PART03_H
2 #define CSCE310HOMEWORK02PART03_H
3
4 #include <vector>
5
6 using namespace std;
7
8 double averageComparisons( vector<int> );
9
10 #endif
```

csce310h0mework02part03.cpp

```
1 /*
2  * Author: Fateh Karan Singh Sandhu
3  * NUID: 17286643
4  *
5  * This function takes an array as an input and returns the average number
6  * of comparisons in a binary search of the array
7  */
8
9 #include "csce310h0mework02part03.h"
10 #include <vector>
```

```

11 #include <iostream>
12 #include <cmath>
13
14 using namespace std;
15
16 double averageComparisons( vector<int> numbers )
17 {
18     //handle edge case for array size 1 and 2
19     if (numbers.size() == 1) {
20         return 1;
21     }
22
23     double averageComparisons = 0.0;
24     int depthOfTree = ceil(log2(numbers.size())); //assume every array is a perfect
BST
25     int countOfLeaves = 0;
26     if (depthOfTree - floor(log2(numbers.size())) == 0) {
27         //if arraysize != 2 and is a perfect log2, add 1 depth for the last node
28         depthOfTree++;
29     }
30     for (int i = 1 ; i < depthOfTree ; i++) {
31         averageComparisons += i * pow(2, i-1); //increment averageComparisons
32         countOfLeaves += pow(2, i-1); //update count of leaves in tree
33     }
34     int leavesAtLast = numbers.size() - countOfLeaves; //count leaves at last level
since BST may not be fully filled
35     averageComparisons += depthOfTree * leavesAtLast; //add comparisons for last level
36
37     return (averageComparisons/numbers.size());
38 }

```