Final Exam

Due Jun 16 at 11:59pm **Points** 41 **Questions** 42
Available until Jun 16 at 11:59pm **Time Limit** 120 Minutes
Allowed Attempts 2

Instructions

Answer the following questions.

This quiz is no longer available as the course has been concluded.

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	7 minutes	39 out of 41
LATEST	Attempt 2	7 minutes	39 out of 41
	Attempt 1	117 minutes	36 out of 41

Correct answers are hidden.

Score for this attempt: **39** out of 41
Submitted Jun 16 at 7:18pm
This attempt took 7 minutes.

Question 1

1 / 1 pts

Consider the following list of numbers in order:

56, 27, 18, 92, 25, 18, 86, 24, 77, 86, 48

How many elements will be studied in linear search, in order to find number 86?

[✓ 7]

6

10

9

Question 2

1 / 1 pts

What is the best, average, and worst runtime complexity for linear search on a list of N items?

best: O(1)
average: O(log N)
worst: O(N)

best: O(log N)
average: O(log N)
worst: O(N)

best: O(log N)

average: O(N)

- worst: O(N * log N)

best: O(1)

average: O(N)

- [✓ worst: O(N)]

Question 3

1 / 1 pts

Consider the following list of numbers in order:

56, 27, 18, 92, 25, 18, 86, 24, 77, 86, 48

How many elements will be studied if we run binary search, in order to find number 86?

- 3
- 2
- 1
- [✓ Binary search does not work in this case.]

Question 4

1 / 1 pts

What is the best, average, and worst runtime complexity for binary search on a sorted list of N items?

- ○ best: O(1)
  average: O(log N)
  worst: O(N)
- [✓ best: O(1)
  average: O(log N)
  worst: O(log N)]
- ○ best: O(1)
  average: O(1)
  worst: O(1)
- ○ best: O(log N)
  average: O(log N)
  worst: O(log N)

Question 5

1 / 1 pts

Consider the following list of numbers in order:

8, 2, 5, 3, 4, 7, 1, 9.

How is the list updated after <u>three iterations</u> of the outermost loop in <u>selection sort</u>?

- ○ 1,2,3,4,5,7,8,9
- ○ 1,2,3,8,5,4,7,9
- ○ 1,2,3,5,8,4,7,9
- [✓ 1,2,3,5,4,7,8,9]

Question 6

1 / 1 pts

Consider the following list of numbers in order:

8, 2, 5, 3, 4, 7, 1, 9.

How is the list updated after <u>three iterations</u> of the outermost loop in <u>insertion sort</u>?

- ○ 1,2,3,4,8,5,7,9
- ○ 2,5,8,3,4,7,1,9

- 1,2,3,8,4,5,7,9
- [✓ 2,3,5,8,4,7,1,9]

Question 7

1 / 1 pts

What is the best runtime complexity for sorting N numbers using insertion sort?

- $O(N^2)$
- O(log N)
- [✓ O(N)]
- O(1)

Question 8

1 / 1 pts

Which case imposes the worst runtime complexity in insertion sort?

- The list is random
- [✓ The list is in descending order]
- The list is in ascending order

Question 9

1 / 1 pts

Consider the following list of numbers in order:

8, 2, 5, 3, 4, 7, 1, 9.

Suppose gap value in shell sort is 4. How many interleaved lists will we have?

- 2
- 8
- [✓ 4]
- 3

Question 10

1 / 1 pts

Which one is correct about shell sort?

- [✓ Shell sort is the same as insertion sort if gap value is 1.]
- On average, insertion sort runs faster than shell sort.
-

Any arbitrary set of gap values can be used to generate sorted lists in shell sort.

Question 11

0 / 0 pts

Consider the following list of numbers in order:

8, 2, 5, 3, 4, 7, 1, 9.

What is the result of partitioning this list in quick sort?

- 2,3,1,85,4,7,9
- 2,3,4,1,8,5,7,9
- 1,2,3,4,5,7,8,9
- [✓ 1,2,3,5,4,7,8,9]

Question 12

1 / 1 pts

What is the worst runtime complexity of partitioning in quick sort?

- O(log N)
- O(N^2)

O(N * log N)

[✓ O(N)]

Question 13

1 / 1 pts

Consider the following list of numbers in order:

8, 2, 5, 3, 4, 7, 1, 9.

Suppose you want to sort this list using merge sort. How many times in aggregate merge sort will be invoked before the whole list is sorted?

7

[✓ 15]

4

3

Question 14

1 / 1 pts

What is the best and worst runtime complexity of merging two lists as part of merge sort?

- ○ best: O(log N)
  worst: O(N)
- ● best: O(N)
  worst: O(N)
- ○ best: O(N * log N)
  worst: O(N * log N)
- ○ best: O(N)
  worst: O(N * log N)

Question 15

1 / 1 pts

Consider the following list of numbers in order:

56, 27, 18, 92, 25, 86, 24, 77, 48

How does the list get updated after the first iteration of the outermost loop in radix sort?

- ○ 18, 24, 25, 27, 48, 56, 77, 86, 92
- ○ 92, 25, 24, 56, 86, 77, 27, 48, 18
- ● 92, 24, 25, 56, 86, 27, 77, 18, 48
- ○ 18, 27, 25, 24, 48, 56, 77, 86, 92

Question 16

1 / 1 pts

Consider the following list of numbers in order:

56, 27, 18, 92, 25, 86, 24, 77, 48

How many iterations are needed in the outermost loop of radix sort to accomplish sorting?

- 3
- 10
- 9
- [✓ 2]

Question 17

1 / 1 pts

In prepending a new node **newNode** into a non-empty singly linked list, which pointers are updated?

- [✓ newNode -> next and then head]
- head -> next and then head
- tail and then newNode -> next

- tail -> next and then tail

Question 18

1 / 1 pts

What is the worst runtime complexity of the following function if the singly-linked list has N items?

```
ListSearch(list, key) {
   curNode = list⇢head
   while (curNode is not null) {
      if (curNode⇢data == key) {
         return curNode
      }
      curNode = curNode⇢next
   }
   return null
}
```

- [✓ O(N)]
- $O(N^2)$
- O(1)
- O(N * log N)

Question 19

1 / 1 pts

Suppose we want to insert node **a** after node **b** in a singly linked list. If **b** is the tail of the list, then how do the pointers need to be updated?

```
a -> next = b -> next;
b -> next = a;
```

```
tail = a;
a -> next = b;
```

[✓
```
b -> next = a;
tail = a;
```
]

```
b -> next = a -> next;
a -> next = b;
```

Question 20

1 / 1 pts

Suppose we want to remove the head of a singly linked list. Which one of the following accomplishes it, assuming that head is not null?

```
if (head -> next != nullptr) {
    head = head -> next;
}
```

```
head = head -> next;
head = head -> next;
if (head == nullptr) {
    tail = nullptr;
}
```

[✓]

Question 21

1 / 1 pts

Suppose we want to remove the node after **a** in a singly linked list. If **a -> next** is not null, then which one of the following accomplishes it?

```
if (a -> next != nullptr) {
    a -> next = a -> next -> next;
}
```

[✓]
```
a -> next = a -> next -> next;
if (a-> next == nullptr) {
    tail = a;
}
```

```
a -> next = a -> next -> next;
```

Incorrect

Question 22

0 / 1 pts

Suppose we have a non-empty doubly linked list. If we insert node **a** after node **b**, how this can be accomplished, assuming that **b** is not the tail of the list?

- [✓]
```
node x = b -> next;
a -> next = b;
a -> prev = x;
b -> next = x;
x -> prev = a;
```

- ○
```
a -> next = b -> next;
a -> prev = b;
b -> next = a;
a -> next -> prev = b;
```

- ○
```
node x = b -> next;
a -> next = x;
a -> prev = b;
b -> next = a;
x -> prev = a;
```

Question 23

1 / 1 pts

Suppose we have a non-empty doubly linked list. If we remove non-head and non-tail node **a** from this list, which pointers need to be updated?

[✓ a -> prev -> next and a -> next -> prev]

a -> next and a -> prev

a -> next -> next and a -> prev -> prev

a -> next -> next and a -> prev -> next

Question 24

1 / 1 pts

Consider running function **f( )** defined below in pseudo-code on a singly-linked list consisting of values 8, 2, 5, 3, 4, 7, 1, 9 in order. What would be the output?

```
f(list) {
   g(list.head);
}

g(node) {
   if (node != null) {
      g(node -> next);
      Print node -> data;
   }
}
```

[✓ 9,1,7,4,3,5,2,8]

9,7,3,2

8,5,4,1

8,2,5,3,4,7,1,9

Question 25

1 / 1 pts

Suppose you use singly-linked lists to implement stacks, where the head of the list represents the top of the stack. In order to implement push, pop, and peek, which list functions are needed, respectively?

- prepend, remove tail, and return tail
- append, remove head, return head
- [✓ prepend, remove head, and return head]
- append, remove tail, and return tail

Question 26

1 / 1 pts

Suppose you use singly-linked lists to implement stacks, where the head of the list represents the top of the stack. If the list consists of the following items in order,

8, 2, 5, 3, 4, 7

specify the content of the stack after the following operations:

push(6);

push(7);

pop();

push(8);

- [✓ 8,6,8,2,5,3,4,7]
- 8,2,5,3,4,7,6,8
- 8,7,6,8,2,5,3,4,7
- 8,7,6,8,2,5,3,4

Question 27

1 / 1 pts

Suppose you use singly-linked lists to implement queues. In order to implement push, pop, and peek, which list functions are needed, respectively?

- append, remove tail, and return tail

- ○ prepend, remove head, and return head
- ○ prepend, remove tail, and return tail
- [✓ append, remove head, and return head]

Question 28

1 / 1 pts

Suppose you use singly-linked lists to implement queues. If the list consists of the following items in order,

8, 2, 5, 3, 4, 7

specify the content of the queue after the following operations:

push(6);

push(7);

pop();

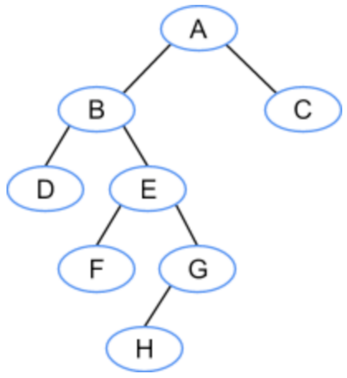
push(8);

- ○ 8,6,8,2,5,3,4,7
- ○ 8,7,6,8,2,5,3,4
- ○ 8,2,5,3,4,7,6,8
- [✓ 2,5,3,4,7,6,7,8]

Question 29

1 / 1 pts

Consider the following binary tree. What is its height?



- 3
- 5
- 2
- [✓ 4]

Question 30

1 / 1 pts

Consider a balanced BST with N nodes. How many nodes need to be studied in the worst case, to find a certain data component?

- O(N)

[✓ O(log N)]

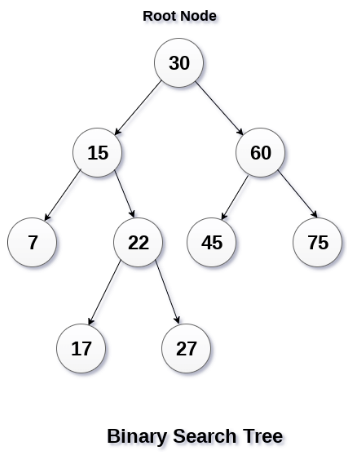
O(N * log N)

O(1)

Question 31

1 / 1 pts

Consider the following BST.



How the tree changes if we insert a node with data 25.

Node 25 is inserted as the right child of node 22, and node 26 becomes the right child of node 25.

Node 25 becomes the left child of node 45.

[✓ Node 25 is inserted as the left child of node 27.]

Node 25 is inserted as the left child of node 30, and node 15 becomes left child of node 25.

Question 32

1 / 1 pts

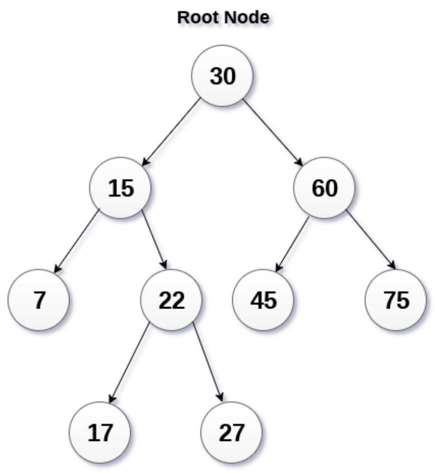
What is the worst runtime for BST insertion if BST has N nodes?

- [✓ O(N)]
- O(log N)
- O(N * log N)
- O(1)

Question 33

1 / 1 pts

Consider the following BST.



Binary Search Tree

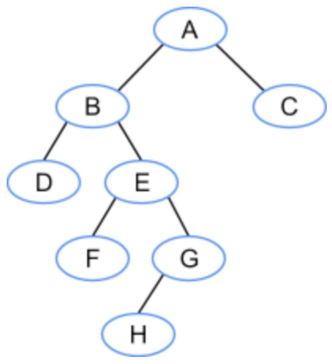
How the tree changes if we remove node 15?

- ○ Node 60 becomes the root, and node 30 becomes left child of node 60.
- ○ Node 22 becomes the left child of node 30.
- ○ Node 27 becomes the left child of node 30.
- [✓ Node 17 becomes the left child of node 30.]

Question 34

1 / 1 pts

Consider the following binary tree.



What is the order of visiting nodes in pre-order traversal?

[✓ A,B,D,E,F,G,H,C]

D,B,F,E,H,G,A,C

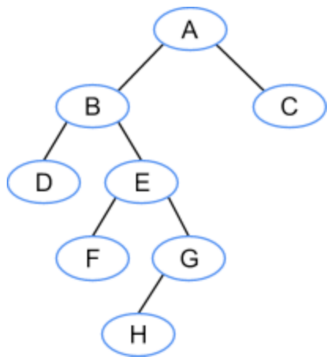
D,F,H,G,E,B,C,A

A,B,C,D,E,F,G,H

Question 35

1 / 1 pts

Consider the following binary tree.



What is the order of visiting nodes in in-order traversal?

- ○ A,B,D,E,F,G,H,C
- ○ D,F,H,G,E,B,C,A
- ○ A,B,C,D,E,F,G,H
- ● D,B,F,E,H,G,A,C

Question 36

1 / 1 pts

Suppose we have a BST with 30 nodes. What is the minimum and maximum height of this tree, respectively?

- ○ 5,30
- ● 4,29

- ○ 5, 29
- ○ 4,30

Question 37

1 / 1 pts

Consider a binary tree T with root X. Assume that node X has left and right subtrees with heights h1 and h2 respectively. What is the height of binary tree T?

- ○ maximum(h1, h2)
- [✓ 1 + maximum(h1,h2)]
- ○ h1 + h2
- ○ 1 + h1 + h2

Question 38

1 / 1 pts

Which one of the following is correct about max heap?

- [✓ A max heap with N nodes has O(log N) height.]
- ○ Order of insertion affects the height of a max heap..
- ○ In a max heap both child nodes are larger than the parent node.

In max heaps, insertion uses percolating down, and removal uses percolating up.

Question 39

1 / 1 pts

Suppose we use an array starting at index 0 to represent a max heap. Consider the node data at index 7. At which indexes do the data components of the left and right children of this node reside?

- 14 and 15
- [✓ 15 and 16]
- 3 and 4
- 4 and 5

Question 40

1 / 1 pts

Consider the max heap represented with the array:

a = {8,6,7,3,4,2,5,1}.

How does the max heap change if we insert a node with data 9?

- {9,6,7,3,4,2,5,1}
- {8,6,7,3,4,2,5,1,9}
- [✓ {9,8,7,6,4,2,5,1,3}]
- {9,8,7,3,4,2,5,1,6}

Question 41

1 / 1 pts

Consider the max heap represented with the array:

a = {8,6,7,3,4,2,5,1}.

How does the max heap change if we remove the root?

- {6,7,3,4,2,5,1}
- {6,3,7,1,4,2,5}
- {7,6,3,4,2,5,1}
- [✓ {7,6,5,3,4,2,1}]

Incorrect

Question 42

0 / 1 pts

Suppose you use max heaps (represented by arrays) to implement priority queues. If the max heap consists of the following items in order,

{6,4,5,2,3}

specify the content of the priority queue after the following operations:

push(9);

push(7);

pop();

push(8);

- {8,6,7,4,5,2,3}
- {8,4,7,2,3,5,6}
- {8,7,6,4,5,2,3}
- [✓ {6,4,5,2,3,9,8}]

Quiz Score: **39** out of 41