METROPOLITAN STATE UNIVERSITY™ OF DENVER
Department of Mathematical and Computer Sciences

CS 2050 - Computer Science II (Spring 2020) I GOT A 94%!

Final Exam Review List[1]

Q1) In the linked list `[4, 10, 2, 8]` which node is the *head* of the list?
a) the node carrying number `4`

Q2) In the linked list `[4, 10, 2, 8]` which node is the *tail* of the list?
d) the node carrying number `8`

Q3) In the linked list `[4, 10, 2, 8]` which node has `index = 2`?
c) the node carrying number `2`

Q4) Suppose `current` refers to a node in a linked list. What statement changes `current` so that it refers to the next node?
c) `current = current.getNext();`

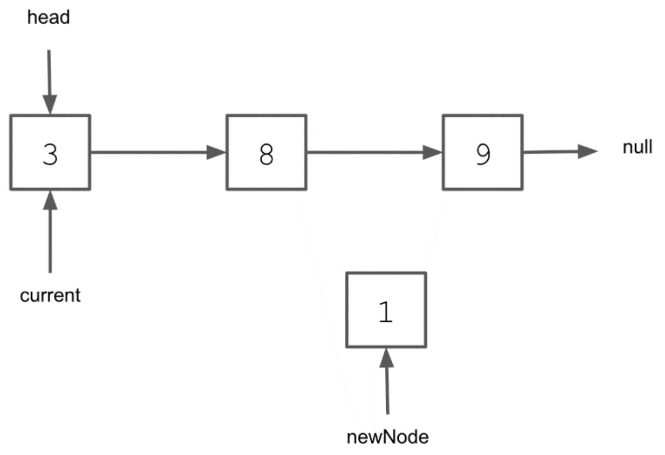
Q5) Suppose `current` refers to a node in a linked list. What boolean expression will be true when `current` refers to the `tail` node of the list?
b) `current.getNext() == null`

Q6) Suppose `p` and `q` refer to nodes in a linked list of integers. What boolean expression indicates whether the numbers in nodes `p` and `q` are the same?
c) `p.getData() == q.getData()`

Q7) Suppose that `p` is a reference variable that contains the `null` reference. What happens at runtime if the program tries to call a method on `p`?
c) `NullPointerException`

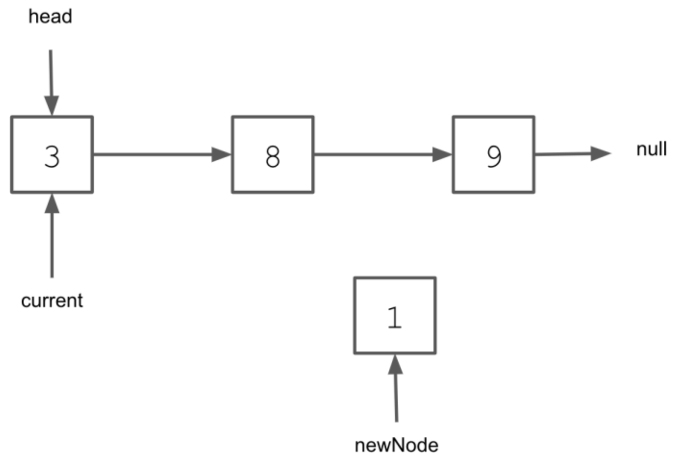
Q8) Consider the linked list `[4, 10, 2, 8]`. If `current` is set to the node carrying number `10`, what node is `current.getNext().getNext()`?
d) the node carrying number `8`

Q9) Consider the linked list and node references below.



Description of the steps to insert the new node between nodes with values `8` and `9`.
b) move `current` to node w/ value `8`, set next of `newNode` to next of `current`, then set next of `current` to `newNode`

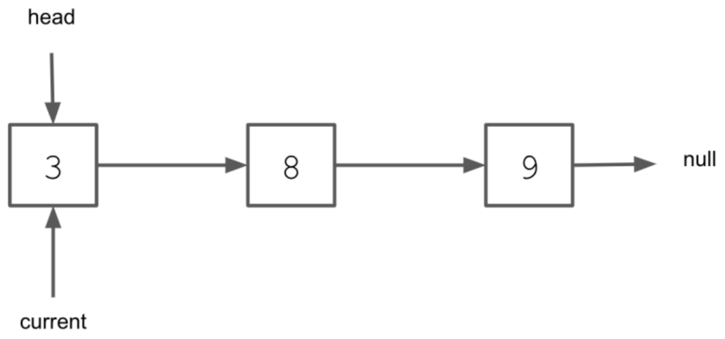
Q10) Consider the linked list and node references below.



Description of the steps to make the new node the head of the list.

a) set next of `newNode` to `head` and then set `head` to `newNode`

Q11) Consider the linked list and node references below.



Description of the steps to remove the node with value 8.
b) assign next of `current`'s next to `temp`; then set next of `current`'s next to `null`; finally set next of `current`

Q12) `lstA` and `lstB` are *singly linked lists* of integers containing the following elements:

`lstA = [3, 8, 2, 9]` and `lstB = [2, 7]`

What is the content of `lstA` and `lstB` after `lstA.doIt(lstB)` is called?

```
void doIt(LinkedList other) {
  Node current = head;
  if (current == null) {
    head = other.head;
    return;
  }
  while (current.getNext() != null)
    current = current.getNext();
  current.setNext(other.head);
```

c) `lstA = [3, 8, 2, 9, 2, 7]` and `lstB = [2, 7]`

Q13) `lstA`, `lstB`, and `lstC` are *singly linked lists* of integers containing the following elements:

`lstA = [3, 8, 2, 9]`, `lstB = [2, 7]`, and `lstC = []`

What is the content of `lstC` after `lstC = lstA.doIt(lstB)` is called? Consider that the `append` method does what its name suggests.

```
LinkedList doIt(final LinkedList other) {
  LinkedList nw = new LinkedList();
  Node calleeCurrent = head;
  while (calleeCurrent != null) {
     nw.append(current.getData());
     calleeCurrent = calleeCurrent.getNext();
  }
  Node otherCurrent = other.head;
  while (otherCurrent != null) {
     calleeCurrent = head;
     boolean found = false;
     while (calleeCurrent != null) {
        if (calleeCurrent.getData() == otherCurrent.getData()) {
           found = true;
           break;
        }
        calleeCurrent = calleeCurrent.getNext();
     }
     if (!found)
        nw.append(otherCurrent.getData());
     otherCurrent = otherCurrent.getNext();
  }
  return nw;
}
```

a) `lstC = [3,8,2,9,7]`

Q14) `lstA` is a *singly linked lists* of integers containing the following elements:

`lstA = [3, 8, 2, 9]`

What is the return value of `lstA.doIt()`?

```
int doIt() {
  Node current = head;
  int s = 0;
  while (current != null) {
     s += current.getData();
     current = current.getNext();
  }
  return s;
}
```

d) 22

Q15) `lstA` is a *singly linked lists* of integers containing the following elements:

`lstA = [32, 32, 32, 32]`

What is the return value of `lstA.doIt(32)`?

```
boolean doIt(int val) {
  boolean flag = true;
  Node current = head;
  while (current != null) {
     if (current.getData() != val) {
        flag = false;
        break;
     }
     current = current.getNext();
  }
  return flag;
}
```

a) true

Q16) `lstA` is a *singly linked lists* of integers containing the following elements:

`lstA = [3, 8, 2, 9]`

What is written in the standard output after a method, defined inside the `LinkedList` class, calls `lstA.doIt(head)`?

```
void doIt(Node current) {
   if(current == null) return;
   System.out.print(current.getData() + " ");
   if(current.getNext() != null)
      doIt(current.getNext());
   System.out.print(current.getData() + " ");
}
```

c) 3 8 2 9 9 2 8 3

Q17) What best describes the purpose of the method `doIt` defined in a `LinkedList` class?

```
int doIt(int data)
{
    Node current = head;
    int var = 0;
    while (current != null)
    {
       if (current.getData() == data)
          return var;
       var++;
       current = current.getNext();
    }
    return -1;
}
```

c) to find and return the position of the given element in the list

Q18) In the worst case, the number of comparisons needed to search a singly linked list of length `n` for a given element is ...

d) n

Q19) In a singly linked list representation, a node contains at least …

c) a data element and a next node reference fields

Q20) The tail of a singly linked list is a node that …

b) has its next reference set to null

Q21) A singly linked list implementation that maintains a reference to the `tail` node the `append` method can be done at …

a) `O(1)` time complexity

Q22) A singly linked list implementation that does NOT maintain a reference to the `tail` node the `append` method can be done at …

c) `O(n)` time complexity

Q23) To build a *singly linked lists* of integers named `lstA` so it would have the following elements:

`lstA = [3, 8, 2]`

Assume `LinkedList<Integer> lstA = new LinkedList<>();` and that `add` and `append` performs a `head` and `tail` insert, respectively. Mark the wrong answer.

a) `lstA.append(3); lstA.append(8); lstA.append(2);`

b) `lstA.add(3); lstA.add(8); lstA.add(2);`

c) `lstA.append(8); lstA.add(3); lstA.append(2);`

d) `lstA.add(8); lstA.append(2); lstA.add(3);`

Q24) What's the output after the statements are executed? Assume that `toString` prints the elements `head` to `tail`.

```
LinkedList<Integer> lstA = new LinkedList<>();
lstA.append(5);
lstA.add(9);
lstA.add(2);
lstA.insert(1, 8);
```

a) [2, 8, 9, 5]

Q25) What's the output after the statements are executed?

```
Stack<Integer> stk = new Stack<>();
stk.push(5);
stk.push(8);
stk.push(2);
stk.pop();
stk.peek();
stk.push(1);
System.out.println(stk);
```

d) (top) 1 8 5 (bottom)

Q26) What's the output after the statements are executed?

```
Queue<Integer> q = new Queue<>();
q.push(5);
q.push(8);
q.push(2);
q.pop();
q.peek();
q.push(1);
System.out.println(q);
```

b) (front) 8 2 1 (rear)

Q27) Consider the following implementation of the `push` method of a stack of integers.

```
void push(int data) {
   Node newNode = new Node(data);
   newNode.setNext(_________);
   top = newNode;
}
```

What would be the correct option to complete the blank?

c) `top`

Q28) Consider an empty stack "s" backed by a static array with a 3 node capacity. What will be `s`'s state after the following sequence of statements are executed?

`s.push(5); s.push(8); s.pop(); s.push(1); s.push(3); s.peek(); s.push(4);`

a) (top) 3 1 5 (bottom)

Q29) Consider an unbounded priority queue "`q`" of strings with a `push` implementation that accepts (in this order) a priority number and a string value. The following sequence of statements are executed:

```
q.push(2, "Janet");
q.push(1, "Bob");
q.push(2, "Sam");
q.push(3, "Anita");
```

What is the state of `q`?
c) `"Anita","Janet","Sam","Bob"`

What is returned from `q.pop()`?
d) `"Anita"`

Q30) What data structure would you use to implement an exhaustive search in order to save memory?
b) stack

Q31) Consider that the *insertion sort* algorithm current state is `[[7, 9, 13, 21] [10]]`. How many comparisons will `insertion sort` do to move `10` to its correct location? (consider ascending order)
b) 3

Q32) Consider that the *insertion sort* algorithm is trying to sort `[7, 9, 10, 13, 21]`. How many comparisons will *insertion sort* do to complete the sorting procedure? (consider ascending order)
a) 4

Q33) Consider that the *selection sort* algorithm is trying to sort `[7, 9, 10, 13, 21]`. How many comparisons will *selection sort* do to complete the sorting procedure? (consider ascending order)
d) 10

Q34) What's the time complexity of the insertion sort algorithm when the input data is already sorted?
c) O(n)

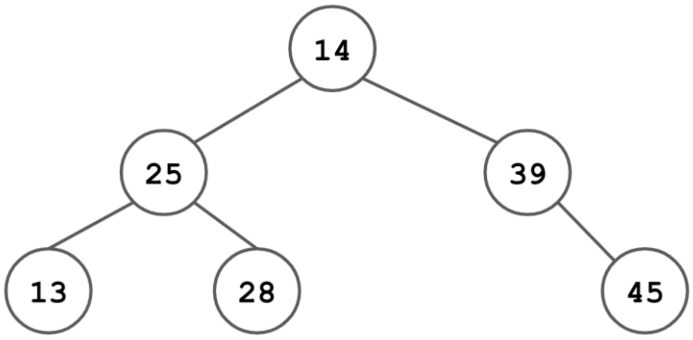
Q35) Consider an implementation of the merge sort algorithm that breaks `[34, 56, 12, 33, 21, 7, 1]` into `[34, 56, 12, 33]` and `[21, 7, 1]`, how many merges will be required to finish the procedure?
c) 6

`[34, 56, 12, 33, 21, 7, 1]`
`[34, 56, 12, 33]` and `[21, 7, 1]`
`[34, 56]` and `[12, 33]` and `[21, 7]` and `[1]`
`[34] [56]` and `[12] [33]` and `[21] [7]` and `[1]`

Q36) If object `a` needs to be placed *after* object `b` in a sorted collection, what should be returned by `a.compareTo(b)` (assume that a and b are of the same type).
c) a number greater than zero

Q37) The *pre-fixed* traversal of a binary search tree is `44, 30, 12, 26, 36, 33, 92, 64, 46, 98`. The *post-fixed* traversal of the same tree is:
b) `26, 12, 33, 36, 30, 46, 64, 98, 92, 44`

Q38) Which node(s) invalidate(s) the following binary search tree in relation to the root node?

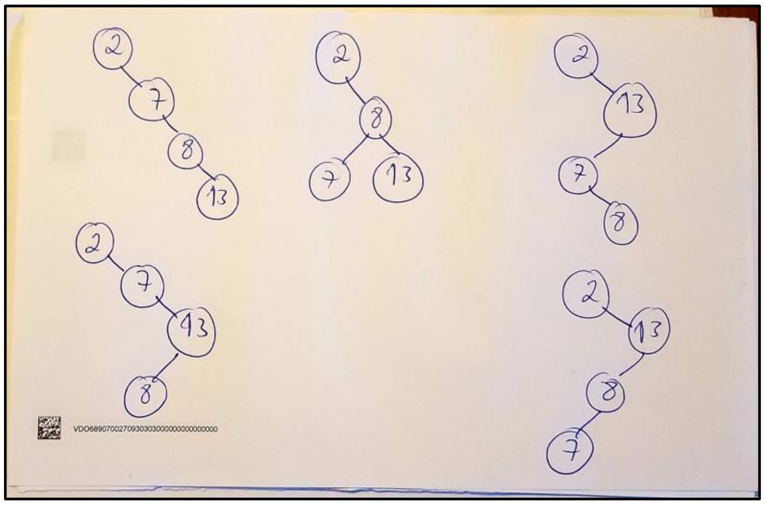


c) 25, 28

Q39) Given a binary search tree, which traversal type would print the values in the nodes in sorted order?
c) in-order

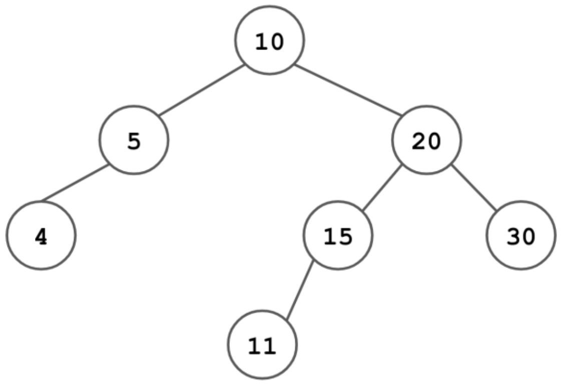
Q40) Which of the following statements about binary trees is NOT true?
a) every binary tree has at least one node

Q41) Using the numbers 2, 7, 8, and 13, how many binary search trees can be built having 2 as the root node?
b) 5



Q42) Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the height of the resulting tree?
b) 5

Q43) Consider the following binary search tree.

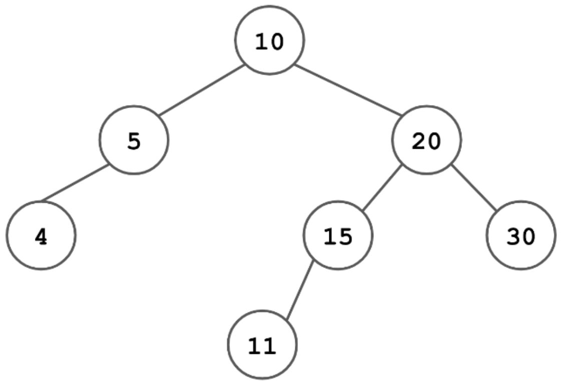


If we randomly search one of the keys present in the above tree, what would be the expected number of comparisons?
b) 2.57

Q44) While inserting the elements 71, 65, 84, 69, 67, 83 in an empty binary search tree in the sequence shown, the element in the lowest level is …
b) 67

Q45) Suppose that we have numbers between 1 and 100 in a binary search tree and want to search for the number 55. Which of the following sequences CANNOT be the sequence of nodes examined?
a) 10, 75, 64, 43, 60, 57, 55
b) 90, 12, 68, 34, 62, 45, 55
c) 9, 85, 47, 68, 43, 57, 55
d) 79, 14, 72, 56, 16, 53, 55

Q46) Consider the following binary search tree.



What would be the pre-order traversal of the tree?
a) 10 5 4 20 15 11 30

Q47) Same tree of Q45. What would be the level-order traversal of the tree?
d) 10 5 20 4 15 30 11

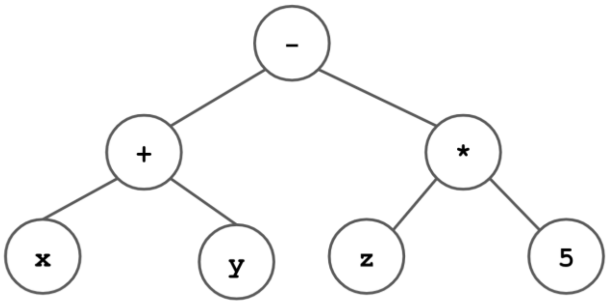
Q48) Same tree of Q45. What would be the post-order traversal of the tree?
c) 4 5 11 15 30 20 10

Q49) Same tree of Q45. What would be the in-order traversal of the tree?
b) 4 5 10 11 15 20 30

Q50) Which order of addition of elements results in the most unbalanced tree?
a) 1 2 3 4 5 6 7 8

Q51) `4 * 3 - x + y` in pre-order notation …
a) `+ - * 4 3 x y`

Q52) Expression that corresponds to the tree below …



c) x + y - z * 5

Q53) Internal nodes of a decision tree represent …
b) relational expressions

Q54) Another name for target values in a decision tree dataset
a) dependent variables

Q55) Decision trees are normally NOT used for …
c) cluster analysis

Q56) A hash table is a data structure ...
b) that maps keys to values

Q57) What is it called when several elements are competing for the same entry in a hash table?
c) collision

Q58) A hash function is a function that …
b) computes the location of the key in an array

Q59) Which of the following is NOT a technique to avoid collisions in a hash table?
b) add elements with collided keys in a linked list

Q60) A hash table of length `10` uses the hash function `k % 10` where `k` is the key. How many collisions will occur when creating the following associations in the given order?
$(3, val_1), (27, val_2), (8, val_3), (83, val_4), (17, val_5), (10, val_6)$
c) 2

Q61) If you have `100` keys values ranging from `1-100` and "`% 16`" as the hash function, what is the probability of collisions to happen?
c) 84%