Name: Stewart Dulaney **ID:** 1545566

Santa Monica College

CS20A Data Structures with C++

Final Exam

Fall 2018

Closed Book, Closed Notes, No Electronic Devices.

Please write your name at least once on every page.

Read each problem carefully, if you do not understand something, ask. If you feel the need to assert any assumptions, as long as they do not contradict anything in the problem description you are free to do so.

Use the provided scratch paper to organize your thoughts. Clearly indicate your final answer on the exam itself.

Good Luck!

Load Factor: $L = \frac{\text{\# of intended entries}}{\text{\# of buckets in table}}$

Closed Hash with Linear Probing: $\text{Ave\#of tries} = \frac{1}{2}\left(1 + \frac{1}{1-L}\right)$

Open Hash: $\text{Ave\#of tries} = 1 + \frac{L}{2}$

Santa Monica College
Spring 2018

CS 20A: Data Structures with C++

Final Exam

Name: Stewart Dulaney **ID:** 1545566

Multiple Choice: Circle the answer that fits best and write out your letter choice into the provided space.

1. __C__ Suppose that items A, B, C, D, and E are pushed, in that order, onto an initially empty stack. Then the stack is popped four times, each time an item as popped it is then inserted into an initially empty queue. If two items are then popped from the queue, what is the next item that will be removed from the queue?

a. A
b. B
[✓ c.] C
d. D
e. E

2. __d__ Consider the following function:

```
int recurse(int a, int b) {
      if (a%b == 2)
            return a;
      else
            return recurse(a + b, a - b);
}
```

What is returned by the call `recurse(5,2)`?

a. 5
b. 7
c. 9
[✓ d.] 10
e. 14

3. __b__ Suppose the following values were inserted into a binary search tree: 6, 7, 10, 20, 15, 24, 8, 27, 30, 31, 33, 42, and 63. How would you classify such a tree's complexity?

a. $O(N^2)$
[✓ b.] $O(N)$
c. $O(\log_2 N)$
d. $O(N \log_2 N)$
e. $O(1)$

~ ordered values results in something that resembles linked list visually and in time complexity

Name: Stewart Dulaney **ID:** 1545566

4. a Consider the following function that takes a pointer to a binary tree:

```
Node * mystery(Node * node) {

        if (node->right == nullptr) return node;
        else return mystery(node->right);
}
```

Which of the following three statements is true:

I. mystery returns the last node processed by an in-order traversal.
~~II.~~ mystery returns the last node processed by a post-order traversal.
~~III.~~ mystery returns the last node processed by a level-order traversal

[✓ a.] I only.
b. II only.
c. III only.
d. I and III only.
e. II and III only.

5. c Which of the following sorting algorithms does not require $O(N^2)$ in the worse possible case?

~~a.~~ insertion sort.
~~b.~~ selection sort.
[✓ c.] merge sort.
~~d.~~ bubble sort.
~~e.~~ quick sort.

6. f With a poorly chosen hash function, it is possible to have a situation in which the search time in that hash table of N items goes as:

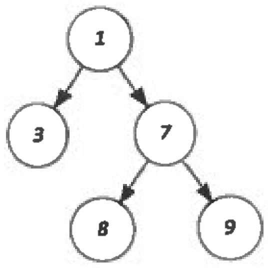
[✓ f.] $O(N)$ closed hash table w/ linear probing that's full
~~g.~~ $O(N!)$
h. $O(N\log_2 N)$
i. $O(N^2)$
~~j.~~ $O(1)$

Name: Stewart Dulaney **ID:** 1545566

7. __d__ In a binary search tree, there is more than one way to delete a node in a tree when that node has two children. One way involves choosing a replacement node from the left subtree. If this is the case, which node are we looking for?

~~a.~~ The smallest node in the subtree.
~~b.~~ The root of the subtree.
c. The node found by traversing all the way to the left.
[✓ d.] The node found by traversing all the way to the right max (left subtree)
~~e.~~ It does not matter, any node will do.

8. __e__ Consider the following binary tree:



Which is of the following statements is correct?

~~a.~~ The tree is a binary search tree.
~~b.~~ The tree is a balanced binary tree. def? → assume balanced means full
~~c.~~ The tree is a min heap. <= children && complete
~~d.~~ The height of the tree is 2. (3)
[✓ e.] None mentioned.

9. __d__ Suppose you want to build an address book app that display entries in alphabetical order by last name and you want fast searching. Which data structure is most appropriate for your implementation?

~~a.~~ Unsorted Linked List
b. Sorted Array
~~c.~~ Stack
[✓ d.] Binary Search Tree
e. Hash table

Insert	Search	ordered	display
	n	no	
n	logn (binary search)	yes	n
logn	log n	yes	n
1	1	no	nlogn (mergeso

Name: Stewart Dulaney **ID:** 1545566

10. c Suppose you want a navigable table of contents for an e-textbook. The book consists of chapters, chapters consist of sections and sections consist of subsections. Which data structure is most appropriate for your implementation?

a. Linked List
b. Hash table
[✓ c.] Tree – hierarchical data
d. Array
e. Heap

11. c Suppose you wanted to develop a task scheduler which assigns tasks based on its importance. Which data structure is most appropriate for your implementation?

a. Linked List
b. Binary Search Tree
[✓ c.] Heap – priority queue
d. Queue
e. Hash table

12. e SMC campus police wants to maintain a database of up to 1800 license plate numbers of people who receive frequent tickets so that it can be quickly determined whether or not a given license plate is in the database. Speed of query is very important; efficient use of memory is also important, but not as important as speed of the query. Which of the following data structures would be most appropriate for this task?

search

~~a.~~ A sorted linked list. n
b. A sorted array with size 1800. log n
c. An open hash table with size 1800. L = 1.0
d. An open hash table with size 3600. L = 0.5
[✓ e.] An open hash table with size 100000. L ≈ 0.02

13. c Suppose you have an Open Hash Table of size 10000. What is the maximum number of entries you can store while maintaining at most 1.25 average number of tries?

a. 1000
b. 3000
[✓ c.] 5000
d. 10000
e. 20000

$$1.25 = 1 + \frac{L}{2}$$
$$0.25 = \frac{L}{2}$$
$$L = 0.50$$

$$0.50 = \frac{\text{\# intended entries}}{10{,}000}$$

5000

Name: Stewart Dulaney **ID:** 1545566

14. During the semester we've examined different ways of implementing a container of objects. For each of the following operations circle the expected time complexity for the indicated data structure containing N items. Your answers should assume "favorable" operating conditions, for example the tree is balanced, hash table has low load factor, there is space in the array, etc.

a. **Search:** returns true or false depending on if the object is in our container.

Sorted array: Assume we use binary search

$O(1)$ [✓ $O(\log_2 N)$] $O(N)$ $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Sorted linked list:

$O(1)$ $O(\log_2 N)$ [✓ $O(N)$] $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Hash table:

[✓ $O(1)$] $O(\log_2 N)$ $O(N)$ $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Binary search tree:

$O(1)$ [✓ $O(\log_2 N)$] $O(N)$ $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

b. **Insert:** add a new item into our container at the correct position.
Assume: analysis based on we know where to insert the item

Unsorted array: Assume we can just put it at the end

[✓ $O(1)$] $O(\log_2 N)$ $O(N)$ $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Sorted array: Assume worst case we have to shift N items if we insert into the 1st positio

$O(1)$ $O(\log_2 N)$ [✓ $O(N)$] $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Hash table:

[✓ $O(1)$] $O(\log_2 N)$ $O(N)$ $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Binary search tree: Assume even though know where to insert item, only hav access to root pointer so still need to traverse wors case log N levels

$O(1)$ [✓ $O(\log_2 N)$] $O(N)$ $O(N\log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Name: Stewart Dulaney **ID:** 1545566

Short Answer:

15. Give the Big-O characterization for each of the following statements with N and P being the input or size of the input.

a. $\log_2 N + \cancel{100000}$ **Answer:** O(log N)

b. $\cancel{N\log_2 N} + \cancel{15}N + \cancel{0.002}N^2$ **Answer:** O (N²)

c. $\cancel{37}N + \cancel{100}N\log_2 N + \cancel{2}\log_2 N$ **Answer:** O(N log N)

$10^3 = 1000$
$2^{10} = 1024$

d. $\cancel{9999}N^3 + \cancel{32N^2} + 2^N$ **Answer:** O(2^N)

e. $N + \underset{N(N-1)/2}{\underset{\shortparallel}{(N-1)}} + (N-2) + ... + 3 + 2 + 1$ **Answer:** O (N²)

f. $N^3 + \cancel{32}P$ **Answer:** O (N³ + P)

g. $\cancel{50}N\log_2 P + \cancel{0.001}P^2$ **Answer:** O(N log P + P²)

h. $\cancel{2^{10}} + \cancel{3^5}$ **Answer:** O (1)

16. Consider a sorted doubly linked list where the first node is the smallest in value, the first node's prev points to nullptr and the last node's next points to nullptr. A variant on this is to make the list circular in that the first node's prev points to the last node in the list and the last node's next points to the first node in the list; this implies that there is no need for a tail pointer. Given this data structure: Assume: analysis on circular sorted doubly linked list

a. What is the Big-O of finding the smallest value in the list?
O(1) head -> value

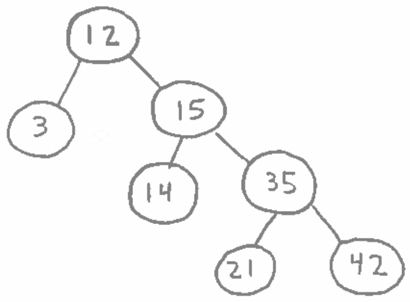
b. What is the Big-O of finding the largest value in the list?
O(1) head -> prev -> value

c. What is the Big-O of determining if a particular value resides in the list? search
O (N)

d. What is the Big-O of inserting a value into the list (not including the cost of finding where it goes)? O(1)

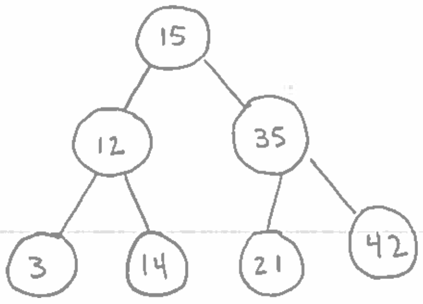
Name: Stewart Dulaney **ID:** 1545566

17. Draw a binary search tree that is created if the following numbers are inserted into the tree in the given order: ~~12~~, ~~15~~, ~~3~~, ~~35~~, ~~21~~, ~~42~~, ~~14~~.



18. Draw a balanced binary search tree containing the same numbers given above.

full



Name: Stewart Dulaney **ID:** 1545566

19. Consider the following binary tree:

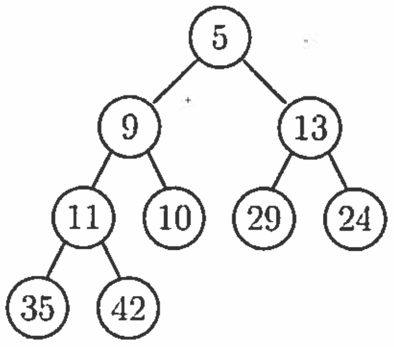


Figure 18

a. Give the pre-order traversal for this tree.

5, 9, 11, 35, 42, 10, 13, 29, 24

b. Give the post-order traversal for this tree.

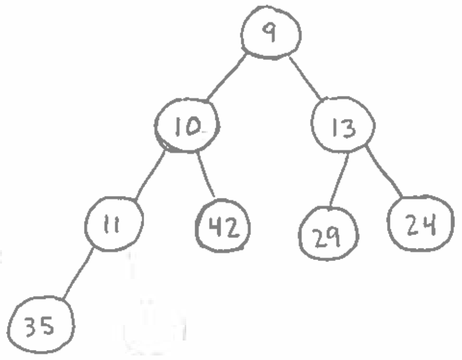
35, 42, 11, 10, 9, 29, 24, 13, 5

c. Give the in-order traversal for this tree.

35, 11, 42, 9, 10, 5, 29, 13, 24

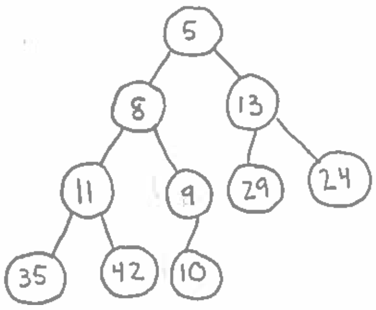
20. You may have noticed that the binary tree in **Figure 18** satisfies the rules for being a min heap,

a. Draw the result of extracting the smallest value from this min heap.



Name: Stewart Dulaney **ID:** 1545566

b. Referring back to the original given min heap shown in **Figure 18**, draw the result after inserting 8.



c. Show the array representation of this min heap shown in **Figure 18**.

5	9	13	11	10	29	24	35	42
0	1	2	3	4	5	6	7	8

d. What is the index for the right child of node 11?

$2(3) + 2 = \boxed{8}$

left child $2i + 1$
right child $2i + 2$

21. Suppose you have a hash table of size 10, with a hash function defined as h(k) = k % 10. We want to insert the values ~~31~~, ~~3~~, ~~44~~, ~~67~~, ~~64~~, ~~4~~, ~~26~~, in the order given.

a. Show the state of the hash table if it is implemented as a closed hash table using linear probing.

	31		3	44	64	4	67	26	
0	1	2	3	4	5	6	7	8	9

Name: Stewart Dulaney **ID:** 1545566

b. Referring to the closed hash table implementation, show the state of the table after the removal of the key 44. Discuss the potential issues with this removal. Note that you are not asked to fix anything, just comment on the issue.

	31		3		64	4	67	26	
0	1	2	3	4	5	6	7	8	9

One issue is that if we search for 64 or 4, the search algorithm presented in class will return false even though 64 and 4 are both in our hash table. This is because the algorithm first hashes to the bucket where it expects to find a value then returns false if it encounters an empty bucket. The lesson is to avoid closed hash tables if you intend to delete values from your container.

c. Suppose we have a hash table that stores strings and we use a hash function that computes buckets using only the length of the input string. Is this a good hash function? Discuss why or why not?

This is not a good hash function b/c it results in many collisions. For example, "cat", "bat", "act", "bee" all hash to the same value, along with all other strings of length 3. One hash function that would result in less collisions would be to compute the sum of the ASII characters in the string, weighted by their index + 1.

Name: Stewart Dulaney		**ID:** 1545566

22. Below is code segment for a doubly linked list class that stores integers. The first node in the list has nullptr for its prev, and the last node in the list has nullptr for its next.

```
class linkedlist {
public:
        ...
        void eraseSecondToLast();
private:
        struct Node {
                int value;
                Node *next;
                Node *prev;
        };
        Node *head;
};
```

The eraseSecondToLast function deletes one node just before the last node in the list with at least three nodes. You may assume that it will be called only for lists with at least three nodes. Below is an incomplete implementation for eraseSecondToLast, complete the implementation with only the options given, duplication of choices may occur. You may not include any additional lines or blanks.

```
void LinkedList::eraseSecondToLast() {

        Node *curr = __head__;
                      1

        while (__curr->next__ != __nullptr__)
                     2                3

             __curr__ = __curr->next__;
                4              5

        Node *temp = __curr->prev__;
                           6

        __curr->prev->prev->next__ = __curr__;
                  7                   8

        __curr->prev__ = __curr->prev->prev__;
              9                  10

        delete __temp__;
                  11
}
```

```
A. curr
B. curr->next
C. curr->next->next->next
D. curr->next->next->prev
E. curr->prev
F. curr->prev->next
G. curr->prev->prev
H. curr->prev->prev->next
I. head
J. head->next
K. nullptr
L. temp
M. temp->next->next
```

Note that the blanks above are for your convenience I will not look at those. **Write the letters corresponding to the filled-in code below:**

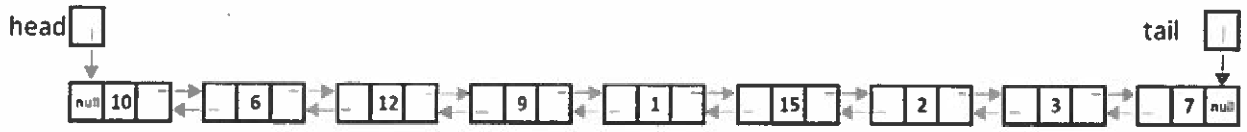
1	2	3	4	5	6	7	8	9	10	11
I	B	K	A	B	E	H	A	E	G	L

Name: Stewart Dulaney **ID:** 1545566

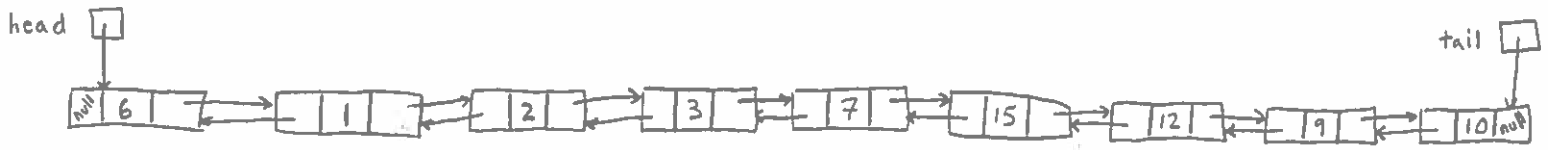
23. Consider the function below that takes pointers to nodes in a doubly linked list. This function calls a swap function, you may assume that swap does what is intended.

```
Node* thingy(Node *s, Node *e) {
        int x = e->value;

        Node *curr = s->prev;

        for (Node *p = s; p != e; p = p->next) {
                if (p->value <= x) {
                        if (curr == nullptr)
                                curr = s;
                        else
                                curr = curr->next;

                        swap(curr->value, p->value);
                }
        }
        if (curr == nullptr)
                curr = s;
        else
                curr = curr->next;

        swap(curr->value, e->value);
        return curr;
}
```

Suppose you have a doubly linked list with the values 10, 6, 12, 9, 1, 15, 2, 3, 7, which can be visualized as:



a. What is the state of the linked list after the function call thingy(head, tail)?



b. What is the Big-O of thingy?

O(N)

Note that thingy is a partition function that can be used to implement QuickSort.

Name: Stewart Dulaney **ID:** 1545566

c. Now suppose you have another function that calls thingy, given below:

```
void majig(Node* l,  Node *h) {

      if (h == nullptr || l == h || l == h->next)
            return;

      Node *p = thingy(l, h);
      majig(l, p->prev);
      majig(p->next, h);
}
```

Discuss the Big-O of majig, including any special circumstances that might result in different complexities.

Note that majig is an implementation of QuickSort that uses thingy as a partition function. The Big-O of majig is O(N log N) under "favorable" operating conditions. In unfavorable conditions, it has a Big-O of $O(N^2)$. These conditions are when the linked list it's given as input is mostly/already sorted or in reverse order. The Big-O changes in these cases b/c due to always choosing the value in the last node as the partition value, the problem size is only decreased by 1 in each recursive step instead of being cut approximately in half. In other words, the partition function thingy, which is O(N), is called approximately N times instead of log N times.