F19 CS20A Final Version B

ALARCON PEDRO ANTONI

TOTAL POINTS

49 / 56.5

QUESTION 1

1 MC 1 **1 / 1**

✓ **- 0 pts Correct:d**

- 1 pts Incorrect

QUESTION 2

2 MC 2 **1 / 1**

✓ **- 0 pts Correct: d**

- 1 pts Incorrect

QUESTION 3

3 MC 3 **1 / 1**

✓ **- 0 pts Correct: c**

- 1 pts Incorrect

QUESTION 4

4 MC 4 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 5

5 MC 5 **0 / 1**

- 0 pts Correct: c

✓ **- 1 pts Incorrect**

QUESTION 6

6 MC 6 **1 / 1**

✓ **- 0 pts Correct: c**

- 1 pts Incorrect

QUESTION 7

7 MC 7 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 8

8 MC 8 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 9

9 MC 9 **1 / 1**

✓ **- 0 pts Correct: a**

- 1 pts Incorrect

QUESTION 10

10 MC 10 **1 / 1**

✓ **- 0 pts Correct: d**

- 1 pts Incorrect

QUESTION 11

11 MC 11 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 12

12 MC 12 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 13

13 MC 13 **1 / 1**

✓ **- 0 pts Correct: e**

- 1 pts Incorrect

QUESTION 14

14 MC 14 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 15

4 pts

15.1 a 0.5 / 0.5

✓ - 0 pts Correct O(P^2)

- 0.5 pts Incorrect

15.2 b 0.5 / 0.5

✓ - 0 pts Correct: O(2^N)

- 0.5 pts Incorrect

15.3 c 0.5 / 0.5

✓ - 0 pts Correct: O(N Log N)

- 0.5 pts Incorrect

15.4 d 0.5 / 0.5

✓ - 0 pts Correct: O(P Log N + N^2)

- 0.5 pts Incorrect

15.5 e 0.5 / 0.5

✓ - 0 pts Correct O(N)

- 0.5 pts Incorrect

15.6 f 0.5 / 0.5

✓ - 0 pts Correct O(N^3 + P)

- 0.5 pts Incorrect

15.7 g 0.5 / 0.5

✓ - 0 pts Correct: O(N Log P + P^2)

- 0.5 pts Incorrect

15.8 h 0.5 / 0.5

✓ - 0 pts Correct O(1)

- 0.5 pts Incorrect

QUESTION 16

4.5 pts

16.1 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.2 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.3 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.4 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.5 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.6 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.7 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.8 0.5 / 0.5

✓ - 0 pts Correct

- 0.5 pts Incorrect

16.9 0 / 0.5

- 0 pts Correct

✓ - 0.5 pts Incorrect

QUESTION 17

7 pts

17.1 0 / 2

- 0 pts Correct: 14 7 3 11 13 16 34 45 18

- 1 pts A few misplaced values

✓ - 2 pts Too many misplaced values

17.2 1 / 1

✓ - 0 pts Correct

- 1 pts Incorrect

17.3 2 / 2

✓ - 0 pts Correct

- **0.5 pts** N Log N (Average case)
- **0.5 pts** Average case on randomly distributed data sets (N Log N)
- **0.5 pts** N^2 (Worse Case)
- **0.5 pts** Worse case on ordered or reverse ordered data sets (N^2)
- **2 pts** Incorrect

17.4 2 / 2

✓ **- 0 pts Correct**
- **1 pts** O(N^2)
- **1 pts** O(N) ordered
- **1 pts** Incorrect reasoning
- **2 pts** Incorrect

QUESTION 18

11 pts

18.1 1 / 1

✓ **- 0 pts Correct: A or B**
- **1 pts** Incorrect

18.2 1 / 1

✓ **- 0 pts Correct: I**
- **1 pts** Incorrect

18.3 1 / 1

✓ **- 0 pts Correct: M**
- **1 pts** Incorrect

18.4 1 / 1

✓ **- 0 pts Correct: C**
- **1 pts** Incorrect

18.5 1 / 1

✓ **- 0 pts Correct: H**
- **1 pts** Incorrect

18.6 1 / 1

✓ **- 0 pts Correct: D**
- **1 pts** Incorrect

18.7 1 / 1

✓ **- 0 pts Correct: Valid Permutations for (7,8,9,10) = {(G,C,D,F), (G,E,D,F),(D,F,E,C)}. D,F,G,C results in a memory leak.Correct**
- **1 pts** Incorrect
- **0.5 pts** D,F,G,C: Memory leak, only points will be deducted from 7,8 but not 9,10.

18.8 1 / 1

✓ **- 0 pts Correct: Valid Permutations for (7,8,9,10) = {(G,C,D,F), (G,E,D,F),(D,F,E,C)}. D,F,G,C results in a memory leak.**
- **1 pts** Incorrect
- **0.5 pts** D,F,G,C: Memory leak, only points will be deducted from 7,8 but not 9,10.

18.9 1 / 1

✓ **- 0 pts Correct: Valid Permutations for (7,8,9,10) = {(G,C,D,F), (G,E,D,F),(D,F,E,C)}. D,F,G,C results in a memory leak.**
- **1 pts** Incorrect

18.10 1 / 1

✓ **- 0 pts Correct: Valid Permutations for (7,8,9,10) = {(G,C,D,F), (G,E,D,F),(D,F,E,C)}. D,F,G,C results in a memory leak.**
- **1 pts** Incorrect

18.11 1 / 1

✓ **- 0 pts Correct: K**
- **1 pts** Incorrect

QUESTION 19

19 0 / 4

- **0 pts** Correct
- **2 pts** Left Subtree (HIM)
- **2 pts** Right Subtree (Sedusa)
- **1 pts** Invalid BST
- **1 pts** Missing nodes
- **1 pts** Erroneous rearragnements

✓ **- 4 pts Incorrect**

QUESTION 20

3 pts

20.1 1 / 1

✓ **- 0 pts Correct: 64 40 33 10 26 29 54 42 27**

- 0.5 pts Minor mistakes

- 1 pts Incorrect

20.2 1 / 1

✓ **- 0 pts Correct: 10 26 33 29 40 42 27 54 64**

- 1 pts Incorrect

20.3 1 / 1

✓ **- 0 pts Correct 10 33 26 40 29 64 42 54 27**

- 0.5 pts Minor mistakes

- 1 pts Incorrect

QUESTION 21

4 pts

21.1 1 / 1

✓ **- 0 pts Correct**

- 1 pts Incorrect

21.2 1 / 1

✓ **- 0 pts Correct**

- 1 pts Incorrect

- 0.5 pts Does not look complete

- 0.5 pts Not valid max heap

21.3 1 / 1

✓ **- 0 pts Correct: 64 40 54 33 29 42 27 10 26**

- 1 pts Incorrect

- 0.5 pts Used the incorrect figure

21.4 1 / 1

✓ **- 0 pts Correct: 10**

- 1 pts Incorrect

QUESTION 22

5 pts

22.1 2 / 2

✓ **- 0 pts Correct**

- 1 pts A few misplaced values

- 2 pts Incorrect

22.2 1 / 1

✓ **- 0 pts Correct: False negatives for searching on keys where shifted due to collisions**

- 1 pts Incorrect/Unclear

22.3 2 / 2

✓ **- 0 pts Correct: A better hash function will likely reduce collisions. Specifically, this function seems to hash on string length, for which there are many collisions. Virtually, any change here will likely yield better results, in particular the approached discussed in lecture where we weight each character. Correct**

- 1 pts Incomplete or Unclear. Any fully correct answer must include a discussion of the hash code. Load factor or table size alone is not sufficient.

- 2 pts Incorrect

Name: PEDRO MARCOS ID: 1722260

Santa Monica College

CS20A Data Structures with C++

Final Exam B

Fall 19

Closed Book, Closed Notes, No Electronic Devices other than a scientific calculator.

**Please write your name on this front page,
then your initials once on each sheet.**

**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: $\text{Ave\#of tries} = \frac{1}{2}\left(1+\frac{1}{1-L}\right)$

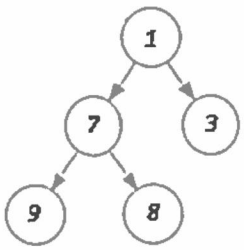
Open Hash: $\text{Ave\#of tries} = 1+\frac{L}{2}$

$Left(i) = 2i+1, Right(i) = 2i+2, Parent(i) = (i-1)/2$

Name: P.A. **ID:** 1722160

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

1. d. Consider the following binary tree:



Which is of the following statements is correct?

a. The height of the tree is 2. → 3
b. The tree is a max heap.
c. The tree is a binary search tree.
d. The tree is a complete binary tree.
e. None mentioned.

4 % 3 = 1

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(4,3)`?

a. 4
b. 5
c. 7
d. 8
e. 10

recurse(7, 1)
recurse(8, 6)

8 % 6 = 2

Name: P. A. ID: 1722260

3. 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 three 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

4. b Suppose you have a Binary Search Tree and it takes O(N) to find the minimum value in the tree, which of the following could be the insertion order of this Binary Search Tree?

a. 7, 19, 8, 28, 16, 30, 32
b. 32, 24 30, 16, 28, 8, 7
c. 7, 8, 16, 24, 30, 28, 32,
d. 24, 8, 30, 7, 16, 28, 32

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

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

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

Which of the following three statements is true:

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

a. III only.
b. II only.
c. I only.
d. I and III only.
e. II and III only.

Name: P. A. ID: 1722260

6. 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 1.2 average number of tries?

a. 20000
b. 10000
c. 4000
d. 3000
e. 1000

7. B Which of the following sorting algorithms is unstable?

a. Merge Sort.
b. Quick sort.
c. Bubble sort.
d. Insertion sort.
e. None mentioned.

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

a. Selection sort.
b. Merge sort.
c. Quick sort.
d. Bubble sort.
e. Insertion sort.

9. A 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:

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

Name: P. A. ID: 1722260

10. d SMC campus police wants to maintain a database of up to 1200 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?

lookup

a. A sorted linked list. → O(N) Hash Table lookup → O(1)
b. A sorted array with size 1200. → O(log₂ N)
c. An open hash table with size 1200.
d. An open hash table with size 2400. → decent load factor and memory usage
e. An open hash table with size 120000.

11. B 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 right subtree. Which node are we looking for from the right subtree?

a. The root of the subtree.
b. The node found by traversing all the way to the left.
c. The node found by traversing all the way to the right → same thing?
d. The smallest node in the subtree.
e. It does not matter, any node will do.

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

a. Hash table → needs to be ordered
b. Binary Search Tree
c. Stack
d. Unsorted Linked List → O(N) search
e. Sorted Array → O(N) insert

13. E 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?

Hierachy

a. Hash table
b. Array
c. Heap
d. Linked List
e. Tree

Name: P. A. ID: 1722260

14. B 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. Queue
b. Heap priority queue
c. Linked List
d. Binary Search Tree
e. Hash table

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. $P+(P-1)+(P-2)+\ldots+3+2+1$ **Answer:** $O(P^2)$

b. $N^3+3200N^2+2^N$ **Answer:** $O(2^N)$

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

d. $P\log_2 N+15N+0.002N^2$ **Answer:** $O(N^2+P\log_2 N)$

e. $\log_2 N+100000N$ **Answer:** $O(N)$

f. N^3+32P **Answer:** $O(N^3+P)$

g. $50N\log_2 P+0.001P^2$ **Answer:** $O(P^2+N\log_2 P)$

h. 10^2+2^3 **Answer:** $O(1)$

Name: P. A. ID: 1722260

16. 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 linked list:

$O(1)$ $O(\log_2 N)$ **$O(N)$** $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Sorted array: Binary Search

$O(\log_2 N)$ $O(N)$ $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Binary search tree:

$O(\log_2 N)$ $O(N)$ $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Hash table:

$O(1)$ $O(\log_2 N)$ $O(N)$ $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

b. **Insert**: add a new item into our container at the correct position.

Sorted array: shifts N elements

$O(1)$ $O(\log_2 N)$ **$O(N)$** $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Unsorted array: → insert @ end

$O(1)$ $O(\log_2 N)$ $O(N)$ $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

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:

$O(1)$ **$O(\log_2 N)$** $O(N)$ $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Heap:

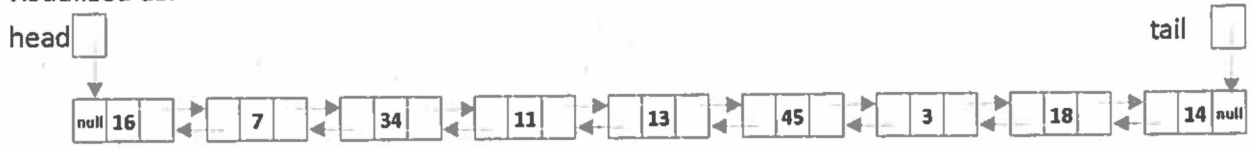
$O(1)$ $O(\log_2 N)$ $O(N)$ $O(N \log_2 N)$ $O(N^2)$ $O(N^3)$ $O(2^N)$

Name: P. A. **ID:** 1722260

17. 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 = e->next;

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

                        swap(curr->value, p->value);
                }
        }
        if (curr == nullptr)
                curr = e;
        else
                curr = curr->prev;

        swap(curr->value, s->value);

        return curr;
}
```

Suppose you have a doubly linked list with the values 16, 7, 34, 11, 13, 45, 3, 18, 14, which can be visualized as:



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

Name: P. A. ID: 1722260

b. What is the Big-O of thingy?

Note that this is a partition function for a QuickSort algorithm. O(N)

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.

- Majig aka QuickSort algorithm sorts the linked based list by recursively swapping all elements less than the partition to the left and the greater elements to the right. The Best Time Complexity occurs when elements are completely unsorted, which gives $O(N \log_2 N)$.

- Worst Time Complexity occurs in already sorted or reversely sorted list. In this scenario, each recursive subset of list eliminates only a single element, which necessitates N calls of majig, ∴ time complexity becomes $O(N^2)$

Name: P. A. ID: 1722260

d. suppose you have another function, given below:

```
void mabob(Node *s) {
        bool swapped = false;
        Node *p;

        if (s == nullptr)
                return;

        do {
                swapped = false;
                p = s;

                while (p->next != nullptr) {
                        if (p->value > p->next->value) {
                                swap(p->value, p->next->value);
                                swapped = true;
                        }
                        p = p->next;
                }
        } while (swapped);
}
```

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

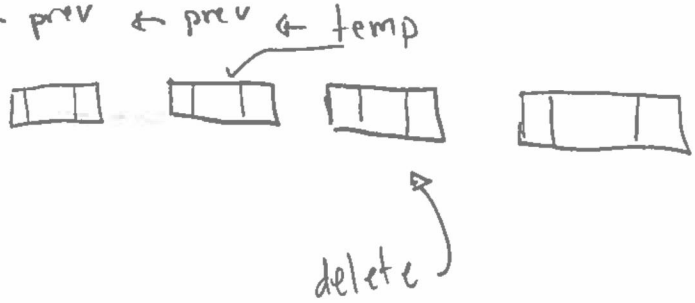
- mabob compares adjacent elements and swaps them if they are not in order. This continues until we hit a nullptr, in other words the end of the list. If there was one swap, the inner while loop runs once more, comparing adjecent elements and swapping if neccessary. (Time Complexity is $O(N^2)$)

- Special Case occurs when no swapping occurs (elements are already ordered) therefore the outter while loop does not run. Time Complexity is $O(N)$

Name: P. A. ID: 1722260

18. 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.

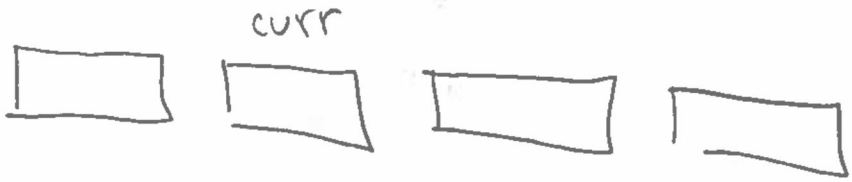
```
class linkedlist {
public:
        ...
        void removeThird ();
private:
        struct Node {
                int value;
                Node *next;
                Node *prev;
        };
        Node *head;
        Node *tail;
};
```



The removeThird function deletes the third node in a list with atleast four nodes. You may assume that it will be called only for lists with at least four nodes. Below is an incomplete implementation for removeThird, complete the implementation with only the options given, duplication of choices may occur. You may not include any additional lines or blanks.

```
void LinkedList::removeThird() {

        Node *curr = ____A____;
                       1

        while (____I____ != ____M____)
                      2            3

               ____C____ = ____H____;
                    4           5

        Node *temp = ____D____;
                        6

        ____G____ = ____C____;
           7           8

        ____D____ = ____F____;
           9           10

        delete ____K____;
                   11

}
```

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



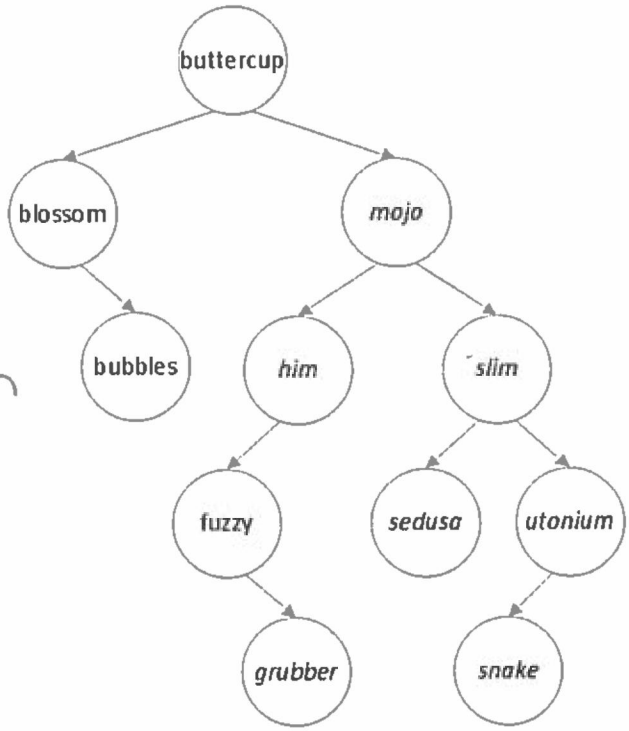
Note that the blanks above are for your convenience I will not look at those. **Write the letters corresponding to the filled-in code below:**

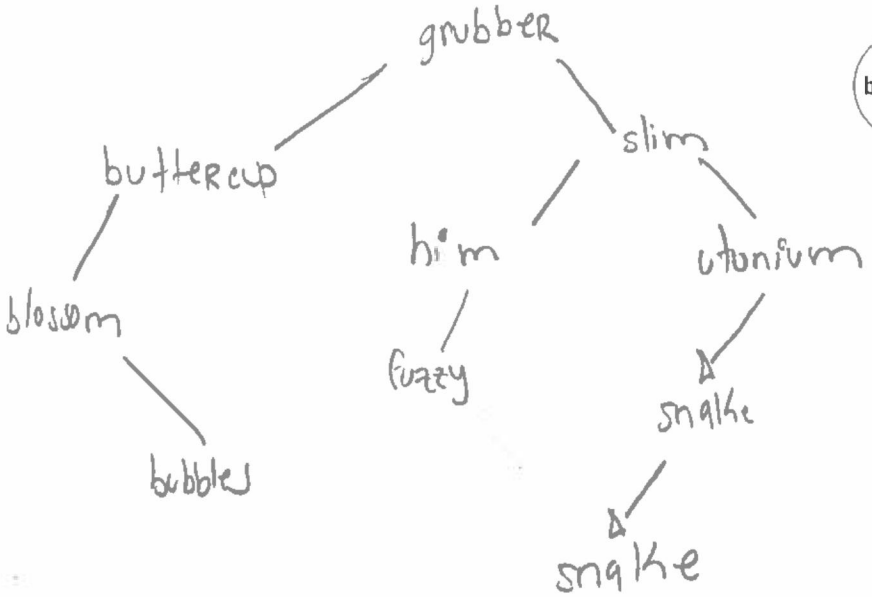
1	2	3	4	5	6	7	8	9	10	11
A	I	M	C	H	D	G	C	D	F	K

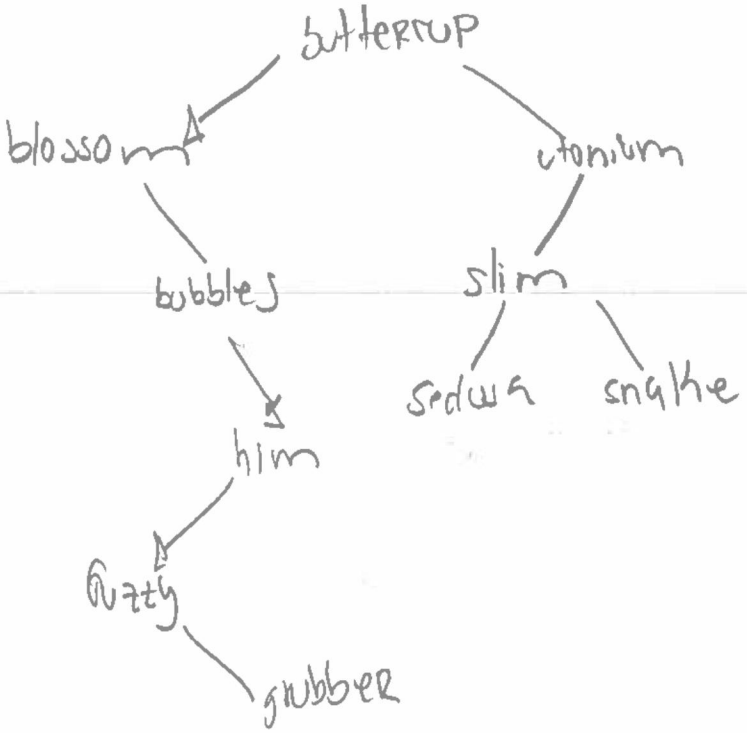
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u

Name: P. A. **ID:** 1722260

19. Consider the following Binary Search Tree. Draw **two** possible resulting trees after deleting "mojo".







Name: P. A. ID: 1722260

20. Consider the following binary tree:

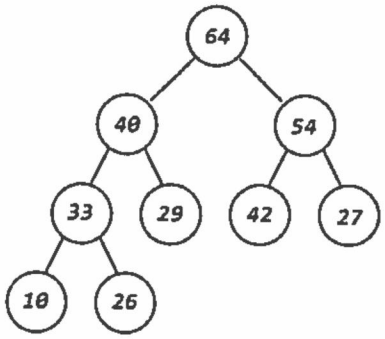


Figure A

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

64, 40, 33, 10, 26, 29, 54, 42, 27

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

10, 26, 33, 29, 40, 42, 27, 54, 64

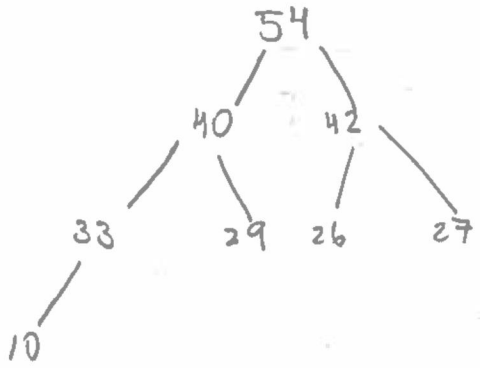
c. Give the in-order traversal for this tree.

10, 33, 26, 40, 29, 64, 42, 54, 27

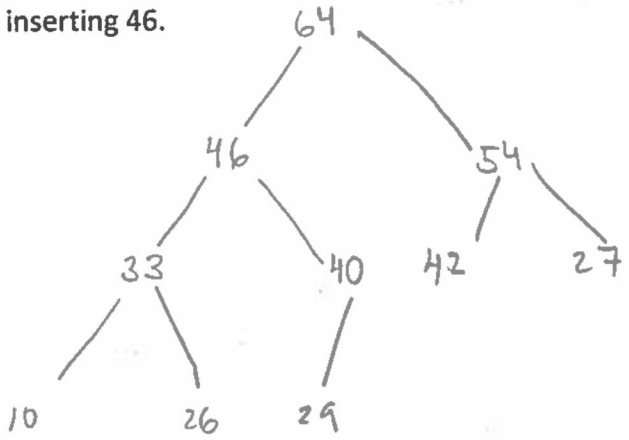
Name: P. A. **ID:** 1722260

21. You may have noticed that the binary tree in **Figure A** satisfies the rules for being a max heap,

a. Draw the result of extracting the largest value from this max heap.



b. Referring back to the original given max heap shown in **Figure A**, draw the result after inserting 46.



c. Show the array representation of this max heap shown in **Figure A**.

$L = 2i + 1$
$R = 2i + 2$

0	1	2	3	4	5	6	7	8
64	40	54	33	29	42	27	10	26

d. From **Figure A**. What is the index for the right child of the node with value 29?

Node w/ value 29 has index 4, therefore

$R = 2(4) + 2 = 8 + 2 = 10$

Right child has index 10

Name: P.A. ID: 1722260

22. 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 6, 21, 48, 56, 37, 68, 36, in the order given.

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

0	68
1	21
2	36
3	
4	
5	
6	6
7	56
8	48
9	37

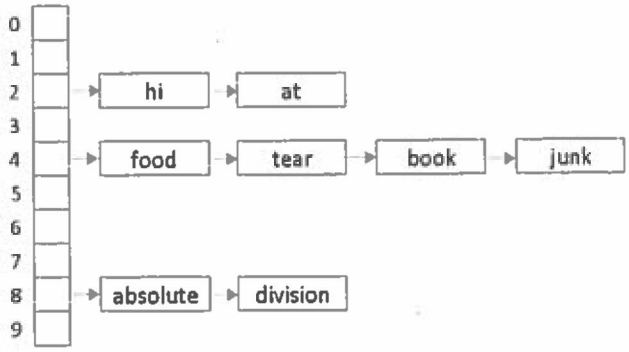
6 % 10 → 6
21 % 10 → 1
48 % 10 → 8
56 % 10 → 6
37 % 10 → 7
68 % 10 → 8
36 % 10 → 6

b. Referring to the closed hash table implementation. Discuss the potential issues with removing the key 6. Note that you are not asked to fix anything, just comment on the issue.

After removing key 6, a search for values 56 and 36 will return false (not found) because search algorithm will find an empty bucket. Therefore, this algorithm will give inaccurate answers.

Name: P. A. ID: 1722260

c. Consider the following open hash table that stores strings. Discuss how we might improve upon this hash table's performance:



Problem:

- It is evident that the hash function maps the keys based on the character count of said key. As displayed in the figure, this causes many collisions which increases operation times.

Possible Solutions:

- The hash function can instead determine the bucket number based on the sum of characters, where each character is given a numerical value based on the ASCII.
- Second, the characters' integer value should be given a weight that is based on position in order to avoid collision w/ words like CAT ↔ TAC