CMSC 341 Exam 1

Nem Negash

TOTAL POINTS

104 / 100

QUESTION 1

1 MC1 3 / 3

✓ **- 0 pts** Correct, (b) post order traversal

- 3 pts Incorrect

QUESTION 2

2 MC2 3 / 3

✓ **- 0 pts** Correct, (c)

- 3 pts Incorrect

QUESTION 3

3 MC3 3 / 3

✓ **- 0 pts** Correct, b

- 3 pts Incorrect

QUESTION 4

4 MC4 3 / 3

✓ **- 0 pts** Correct, a

- 3 pts incorrect

QUESTION 5

5 MC5 3 / 3

✓ **- 0 pts** Correct, b

- 3 pts Incorrect

QUESTION 6

6 MC6 3 / 3

✓ **- 0 pts** Correct, a

- 3 pts Incorrect

QUESTION 7

7 MC7 3 / 3

✓ **- 0 pts** Correct, a

- 3 pts Incorrect

QUESTION 8

8 DS vs ADT 6 / 6

✓ **+ 3 pts** ADT is specification

✓ **+ 3 pts** DS is implementation

+ 3 pts Something vaguely correct

+ 0 pts Incorrect or too vague

QUESTION 9

9 LL insert special cases 8 / 8

✓ **- 0 pts** Correct

- 2 pts One wrong or missing

- 5 pts Two wrong or missing

QUESTION 10

10 Normal BST inserts 9 / 9

✓ **- 0 pts** Correct

- 2 pts 71 < 74

QUESTION 11

11 Awful binary search question 6 / 6

✓ **- 0 pts** This question was unintentionally confusing. The list of items was supposed to be sorted. Full credit if you wrote something either (1) true for the sorted list, (2) sort of true for the supplied list, or (3) true for the tree above.

- 6 pts blank or wholly incorrect

QUESTION 12

12 Big O 1 3 / 3

✓ **- 0 pts** Correct, O(n lg n)

- 2 pts Missing or incorrect justifications

- 1 pts Vague justification

- 3 pts Blank or wholly incorrect. No points for justification are given if runtime is wrong.

QUESTION 13

13 Big O 2 3 / 3

✓ **- 0 pts** **Correct, O(n^2 * lg n)**

- 2 pts Missing or incorrect justifications

- 1 pts Vague justification

- 3 pts Blank or wholly incorrect. No points for justification are given if runtime is wrong.

QUESTION 14

14 Big O 3 **0 / 3**

- 0 pts Correct, O(1)

- 0.5 pts O(3000) is wrong. You should drop the constant coefficient of 3000.

✓ **- 3 pts** **Blank or wholly incorrect. No points for justification are given if runtime is wrong.**

- 2 pts Missing or incorrect justifications

- 1 pts Vague justification

QUESTION 15

15 Inorder traversal **2 / 2**

✓ **- 0 pts** **Correct, 1 5 7 8 9**

- 2 pts Blank or incorrect

- 1 pts Mostly correct

QUESTION 16

16 Postorder traversal **2 / 2**

✓ **- 0 pts** **Correct, 1 5 8 9 7**

- 2 pts Blank or incorrect

- 1 pts Mostly correct

QUESTION 17

17 Descendents **1 / 1**

✓ **- 0 pts** **Correct, 1 5 8 9 in any order**

- 1 pts Incorrect

QUESTION 18

18 Siblings **1 / 1**

✓ **- 0 pts** **Correct, 9**

- 1 pts Incorrect

QUESTION 19

19 Tree judging **5 / 5**

✓ **- 0 pts** **Correct, no no no yes yes**

- 5 pts incorrect or blank

- 1 pts It is not complete

- 1 pts It is not perfect

- 1 pts It is not full

- 1 pts It IS an AVL tree

QUESTION 20

20 Step() **22 / 25**

✓ **+ 7 pts** **Move ptr ahead correctly**

+ 3 pts Throws exception if pointer starts null (if you don't you seg fault)

✓ **+ 5 pts** **Throws exception if pointer reaches null**

✓ **+ 5 pts** **Advances pointer correct number of hops**

✓ **+ 5 pts** **ptr is "returned" by altering it through reference. i.e. no attempt to return pointer, and no temp pointer updated INSTEAD of ptr**

- 2 pts Exception should NOT be caught within this function. The purpose of an exception is to force the calling function to "deal" with a mistake it made.

- 3 pts Segmentation Fault

+ 4 pts Break/return instead of exception

- 5 pts Deleting the pointer is bad

+ 0 pts Blank or incorrect

QUESTION 21

21 Why the funny characters? **5 / 5**

✓ **- 0 pts** **Correct, this is pass by reference, which means that the changes you made to ptr are visible in the calling function.**

- 5 pts Incorrect

QUESTION 22

AVL 0 pts

22.1 Insert 1 **1 / 0**

✓ **+ 1 pts** **Correct**

+ 0 pts Incorrect or blank

22.2 Insert 2 **1 / 0**

✓ **+ 1 pts** **Correct**

+ 0 pts Incorrect, 2 > 1

+ 0 pts Blank

22.3 Insert 5 2 / 0

✓ + **2 pts** Correct

+ **0 pts** Incorrect

22.4 Insert 4 1 / 0

✓ + **1 pts** Correct

+ **0 pts** Incorrect

22.5 Insert 3 2 / 0

✓ + **2 pts** Correct

+ **0 pts** Incorrect

QUESTION 23

23 The Magic 3 / 0

✓ + **3 pts** Correct

CMSC 341 Spring 2020 (Prof. Johnson)
Exam 1

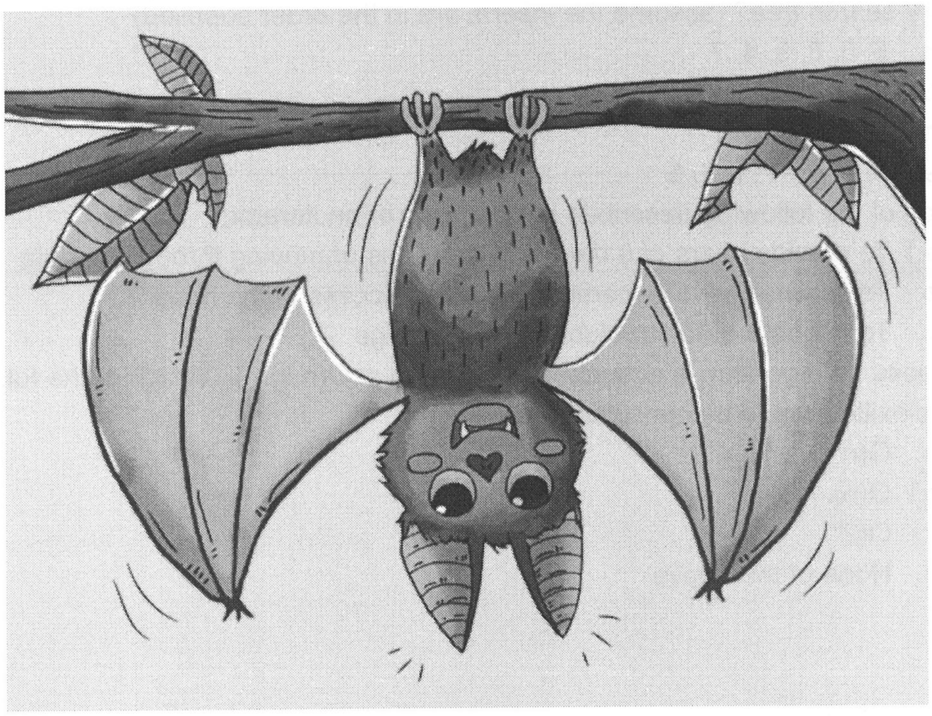
Name: Nem Negash

UMBC email: XK28378 @umbc.edu

DO NOT OPEN YOUR EXAM UNTIL INSTRUCTED!

DO NOT TRUST YOUR GRADE TO A STAPLE: PUT YOUR NAME ON EACH PAGE

Write clearly! Unreadable answers will receive no credit.



Your psychic bat friend knows you're going to do great!

Img cred: freepik

CMSC 341 Exam 1

Name: [illegible] [illegible]

Multiple choice (3 points each, 21 points total)

Circle the answer you choose. If you change your mind, please make it clear what your final answer is.

1. Given a binary tree node, the best way to calculate its height is through a:
   a. Preorder traversal VLR
   [✓ b. Postorder traversal] LRV
   c. Inorder traversal LVR
   d. Reverse inorder traversal
2. Given a binary ***search*** tree node, the way to obtain a list of it and its descendants in ascending key order is:
   a. Preorder traversal
   b. Postorder traversal
   [✓ c. Inorder traversal]
   d. Reverse inorder traversal
3. Which of the following lists of integer insertions creates the TALLEST ***unbalanced*** binary search tree? (assume the inserts are in the order supplied)
   a. 5, 3, 8, 9, 4, 7
   [✓ b. 3, 8, 9, 10]
   c. 6, 2, 1, 3, 9, 7
   d. 3, 0, 6, -1, 1, 7, 5
4. Which of the following describes the purpose of an iterator:
   [✓ a. To provide users of a data type a means of moving through its data]
   b. To allow all private member variables access to the heap
   c. To manage data structure memory usage
5. Suppose an algorithm is described as running in O(n lg n). Which of the following time complexities would be considered ***faster***:
   a. $O(n^2)$
   [✓ b. O(n)]
   c. $O(2^n)$
   d. None of the above

Name: Nem Negash

6. Which of the following pointers will NEVER cause a segmentation fault when they are dereferenced:
   [✓ a.] int * a = new int(3);
   b. int * b = nullptr;
   c. int * c = new int(421);
      delete c;
      c = nullptr;
   d. int * d;
7. The difference between a static data structure and a dynamic data structure is:
   [✓ a.] Static data structures have a fixed capacity, and dynamic data structures do not
   b. Dynamic data structures use only heap memory, whereas static data structures use only stack memory
   c. Data in static data structures may not be changed after being inserted, whereas data in dynamic data structures can be changed

CMSC 341 Exam 1

Name: Nem Nguyen

Short answer (29 points)

11. What is the difference between an abstract data type and a data structure? (6 points)

ADT is the data and it's methods with no form or structure. A data structure is the implementation of ADT is some form.

12. Suppose you are going to write a linked list insert() method that takes the index where the data is going to be inserted. Describe three special cases of insertion that you must handle in your code. YOU DON'T NEED TO WRITE CODE! (8 points)

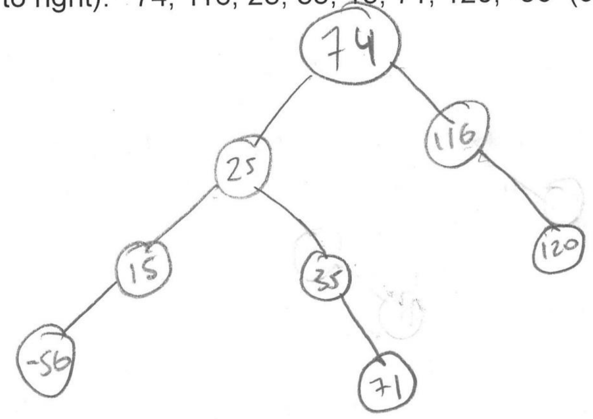
12.1: insert at beginning/end

12.2: if list is empty

12.3: insert in the middle

Name: Nem Negash

13. Draw the (***normal***) binary search tree that results from the following insertions (read from left to right): 74, 116, 25, 35, 15, 71, 120, -56 (9 points)



14. Given the following list [74, 116, 5, 54, 26, 15, -56] Which value for a BINARY search gives: (6 points)

Best case performance 54

Average case performance 116, 15

Worst case performance 74, 5, 26, -56

CMSC 341 Exam 1

Name: Neon Negash

Code analysis (9 points)

For the following questions, give asymptotic worst-case running times for each of the following code fragments. Characterize the running time as closely as possible (*e.g.*, don't say $O(n^3)$ if $O(n^2)$ also works). Express running times as a function of n. Make sure you give a satisfactory **BUT BRIEF** explanation — it is worth a significant fraction of the points.

16. 3 points

Complexity: $O(n \cdot \log n)$

Justification:

first loop runs in n time and second runs on log n since they are nested so it's multiplied

```
for(i = 0; i < n; i++) {
   for(j = 1; j < n; j *= 2) {
     size++;
   }
}
```

17. 3 points

Complexity: $O(n^2 \log n)$

Justification:

first loop runs on n^2 and second loop is log n b/c j is divided by 2 each time they are multplied b/c they are nested

```
for(i = 0; i < n * n; i++) {
   for (j = n; j > 1; j /= 2){
     size++;
   }
}
```

n^2 log n

18. 3 points

Complexity: $O(2^n)$

Justification:

best case it would runs 3000 times but worse case is if $2^n > 3000$ then it runs 2^n b/c num = 2^n

```
int num = 2 ^ n
while (num > 3000) {
   num = 3000
}
for (i = 0; i < num; i++) {
   size++;
}
```

best case = num = 300

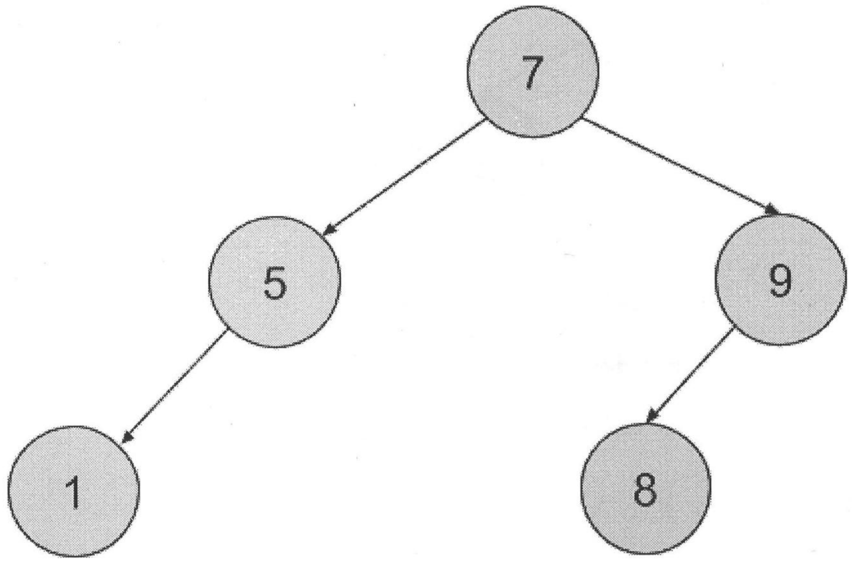
worse case = num > 3000

num = 2^n

Name: Nem Negash

Trees (11 points)

Refer to the following binary search tree for this section:



19. What is the inorder traversal for this tree? (2 points) LVR

1, 5, 7, 8, 9

20. What is the postorder traversal for this tree? (2 points) LRV

1, 5, 8, 9, 7

21. What are the descendent(s) of 7? (1 points)

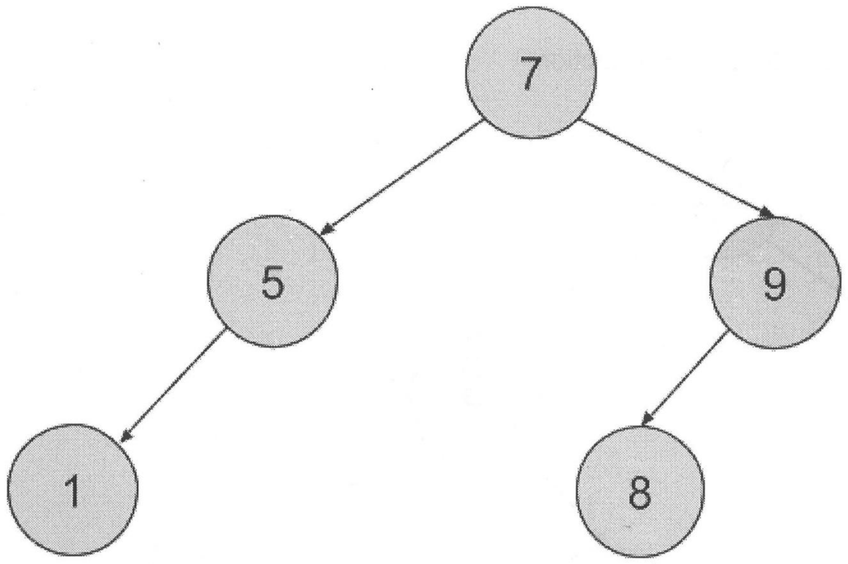
5, 9, 1, 8

22. What is the sibling of 5? (1 points)

9

Name: Prem Nagash

Copy of the previous tree:



23. Is this tree:
    a. Full? (1 point) no
    b. Complete? (1 point) no
    c. Perfect? (1 point) no
    d. A valid BST? (1 point) yes
    e. A valid AVL tree? (1 point) yes

Name: Nem Negash

Coding (30 points)

Another reminder to write clearly. Code should be C++'ish, that is, it does not need to be syntactically perfect, but it should be expressing the right algorithm.

Write the step() function on the next page. It should move the supplied pointer however many supplied spaces FORWARD in the linked list.

You should **throw a range_error exception** if supplied hops would move the ptr beyond the end of the list.

```
class Node {
public:
  Node(int data);

  Node * _next = nullptr;
  int _data = -1;
};

Node::Node(int data) : _data(data) {}

int main() {
  Node * node = new Node(40);
  Node * curr = node;
  for (int i = 0; i < 10; ++i) {
    curr->_next = new Node(i);
    curr = curr->_next;
  }
  curr = node;
  step(curr, 2);
  cout << curr->_data << " <-- should be 1" << endl;
  //I'm not going to delete things to save space for the exam
}
```

40, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Space for coding on the next page

CMSC 341 Exam 1

Name: Nem Negash

```
if(ptr == nullptr){
    throw invalid_argument("list is emp
}
```

Code the step function (25 points)

```
void step(Node *&ptr, int hops) {
  for(int i=0; i<hops; i++){
        ptr = ptr->_next;
        if( ptr == nullptr){
           throw range_error("the supplied hops goes out of range");
        }
  }
}
```

Analysis (5 points)

Sorry I wrote alot :(

Explain ***BRIEFLY*** why ptr in the above function must be of type Node *&ptr.

Since the function is a void function the values that are being changed in the function (in this case ptr) have to be passed by reference, if it wasn't a void pass by refrance wouldn't be required since it can return the new ptr at the end of the function but the user would have to set ptr = to the function call.

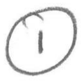
Name: Nem Negash

Extra credit (10 points)

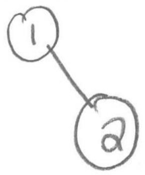
AVL tree insertion (7 points)

Insert [1, 2, 5, 4, 3] one at a time into an AVL Tree.

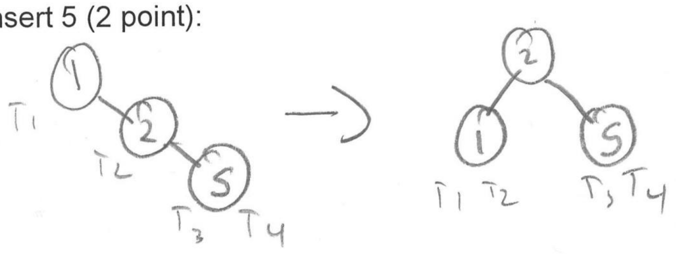
Insert 1 (1 point):



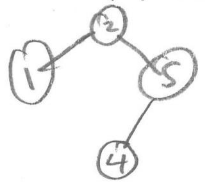
Insert 2 (1 point):



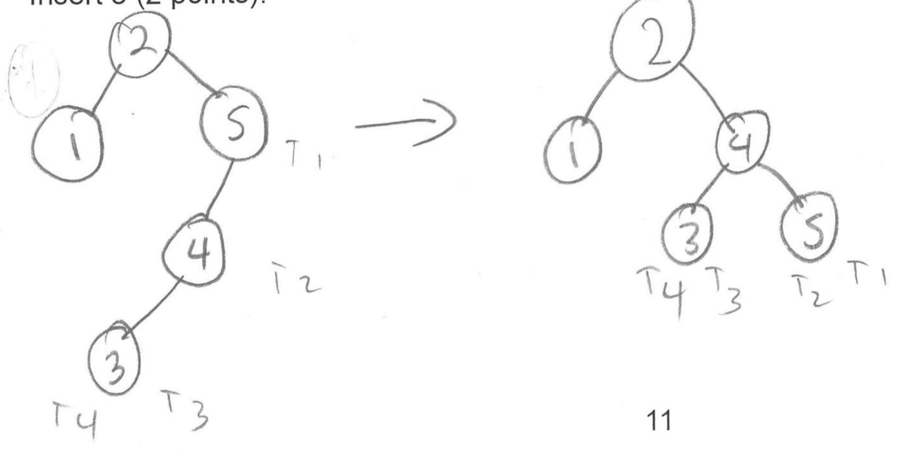
Insert 5 (2 point):



Insert 4 (1 points):



Insert 3 (2 points):



CMSC 341 Exam 1

Name: Nem Negash

The following question is graded for ***participation***, not quality (*i.e.*, you can put something really simple and get full credit). Do not spend time on this question that you could be spending on questions graded for correctness.

Fill the space below with something that expresses your identity. Drawings, quotes, words, anything! Super awesome entries may get hung outside my office. Sign your work if you want credit, otherwise be mysterious and leave it blank. (3 points)

~~[illegible]~~