Midterm 1 - S1

Graded

Student

Sara Huston

Total Points

29.9 / 30 pts

Question 1

Q1(a) **1** / 1 pt

✔ **– 0 pts** Correct

Question 2

Q1(b) **1** / 1 pt

✔ **– 0 pts** Correct

Question 3

Q1c(i) **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect

– 0.5 pts Missing/incorrect calculation

– 0.5 pts Incorrect size of float

Question 4

Q1c(ii) **1** / 1 pt

✔ **– 0 pts** Correct

– 0.5 pts Fail to show the default value

– 1 pt Incorrect

Question 5

Q1c(iii) **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Incorrect/Missing.

Question 6

Q2

5 / 5 pts

- ✔ **+ 0.5 pts** 0.5 pts
- ✔ **+ 0.5 pts** 1 pt
- ✔ **+ 0.5 pts** 1.5 pt
- ✔ **+ 0.5 pts** 2 pt
- ✔ **+ 0.5 pts** 2.5 pt
- ✔ **+ 0.5 pts** 3 pt
- ✔ **+ 0.5 pts** 3.5 pt
- ✔ **+ 0.5 pts** 4 pt
- ✔ **+ 0.5 pts** 4.5 pt
- ✔ **+ 0.5 pts** 5 pt

+ 0 pts Incorrect or Missing

1 Good job!

Question 7

Q3

5 / 5 pts

- ✔ **– 0 pts** Correct

– 5 pts Incorrect or blank

– 0.7 pts Incorrect Head

– 0.7 pts Incorrect Null Pointer

– 2.5 pts Incorrect address length for all nodes

– 0.7 pts Incorrect address length for some nodes

– 3 pts Major error

– 1.5 pts Incorrect address length for addresses

– 1.5 pts Incorrect address length for short integers

Question 8

Q4(a)

2 / 2 pts

✔ **– 0 pts** Correct

– 2 pts Blank or ungradable answer

– 1 pt Incorrect insertion position

– 0.5 pts Previous node not updated to point to new node

– 0.3 pts Minor mistakes

Question 9

Q4(b)

1 / 1 pt

✔ **– 0 pts** Correct

– 0.33 pts Missing one or incorrectly answered one part of the question.

– 1 pt Did not provide a clear or no answer to the question.

– 0.66 pts Missing two or incorrectly answering two parts of the question.

Question 10

Q5(a)

1.9 / 2 pts

– 1 pt Incorrect advantage

– 1 pt Incorrect disadvantage

– 0.5 pts Disadvantage mentions that iterative algorithms are more efficient / recursive are less efficient (not in terms of runtime), but no mention of how it's better in terms of space complexity with less call stacks

– 0.25 pts Advantage somewhat has the right idea that recursion is generally more elegant / generally helps with comprehending a specific class of dividable problems, but not quite all the way there.

✔ **– 0.1 pts** Mention of using more stack memory, but no specific mention of this being specifically due to more call stacks

– 0.2 pts Mention of using more memory / space, but no mention of specifically stack memory w/ more call stacks

– 0.2 pts Correct / partially correct, but there is an incorrect statement about recursive vs iterative time complexities

– 0 pts Correct!

Question 11

Q5(b)

2 / 2 pts

✔ **– 0 pts** Correct

– 1 pt Click here to replace this description.

– 0.5 pts Click here to replace this description.

– 2 pts Click here to replace this description.

Question 12

Q5(c) **2** / 2 pts

✔ **– 0 pts** Correct

– 0.5 pts Click here to replace this description.

– 1 pt Click here to replace this description.

– 2 pts Click here to replace this description.

Question 13

Q6(a) **3** / 3 pts

✔ **– 0 pts** Correct

– 2 pts Best and worst case would both be O(n), since the second (inner) loop runs for a constant number of iterations, independent of N, and the outer loop must run for N

– 1 pt Best and worst case would both be O(n), since the second (inner) loop runs for a constant number of iterations, independent of N, and the outer loop must run for N

– 0.5 pts Time complexities are mostly correct, however, keep in mind for big-O notation, we would not refer to the array a, rather, a variable N to denote the size of a. I.e. instead of O(a), we would write O(n)

– 0.25 pts Correct, but did not explain reasoning

– 1 pt Correct worst-case run time complexity, no best case given

Question 14

Q6(b) **3** / 3 pts

✔ **– 0 pts** Correct

– 1 pt The worst case would actually be O(n^2), this case would occur when all a[i] are > 10

– 1 pt The best case would actually be O(1), the function would return when a[0] happens to be <= 10 in this case

– 0.25 pts Correct, but did not explain reasoning

– 0.5 pts Correct run time complexities, but ensure that you use the variable N for big-O notation, rather than the variable a which refers to the array in this case.

– 1 pt Worst case not provided for method 2, would be O(n^2)

– 0 pts Click here to replace this description.

122

2/12/2025

COMP 210 – Data Structures and Analysis

Spring 2025, Section 1

First Midterm Exam

Date: February 13th, 2025 Student's Full Name: Sara Huston

Marks: 30 Duration: 70 minutes, Pages: 8 Student ID: 730459812

NOTE: PLEASE ANSWER ALL QUESTIONS GIVEN IN THIS QUESTION PAPER IN THE SPACES GIVEN.

Question No.	Points allocated
1	*5*
2	*5*
3	*5*
4	*3*
5	*6*
6	*6*
Total Points	**30**

Question 1 [5 points]

a) Convert binary number 1101 1111 into Hex. Show your workings:

$1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3$ $\quad$ $1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3$

$= 1 + 2 + 4 + 8$ $\quad$ $= 1 + 0 + 4 + 8$

$= 15 = F$ $\quad$ $= 13 = D$

DF

b) Convert "DB" Hex into decimal. Show your working.

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

$D = 13$ $\quad$ $B = 11$

$13 \cdot 16^1 + 11 \cdot 16^0 =$ 219

13 × 16 = 78, 130, 208 + 11 = 219

c) Given an array 'a' defined as: **float[] a = new float[10];**

i) Assume that the base address of a is 1600 (decimal).

What would be the address of a[7] in decimal? (show your calculation) (range)

$a[7] = 1600 + 7 \cdot 4$ $\quad$ 1628-1631

$= 1600 + 28 =$ 1628 (base address)

ii) What would be the result of executing the following statement after the array is defined?

```
System.out.println(a[9]);
```

0.0

iii) What would happen if we subsequently had the following statement?

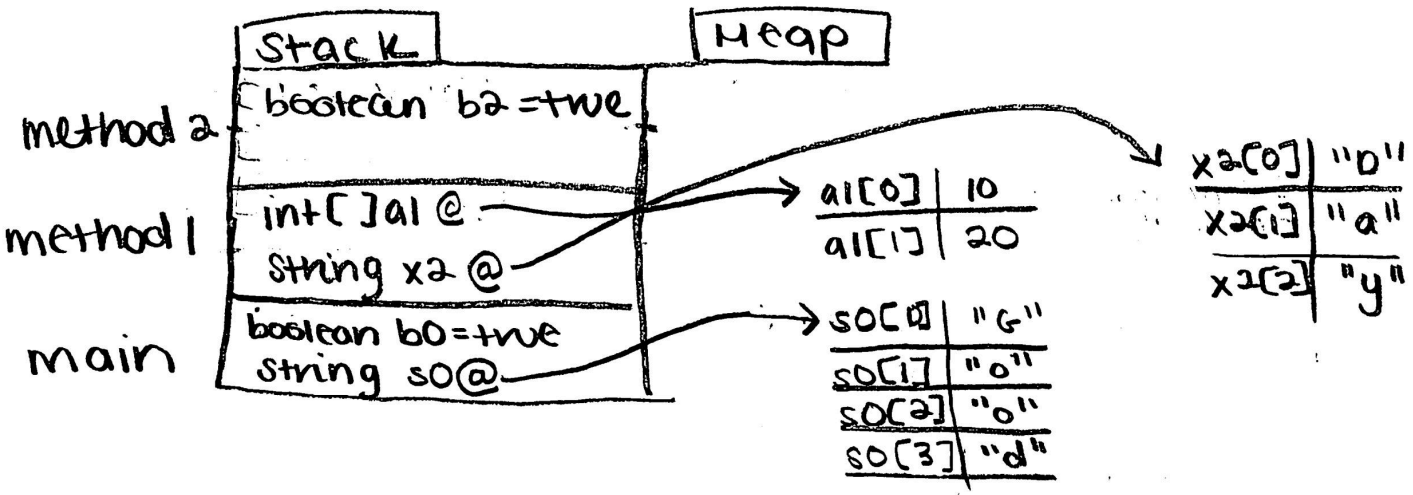
int[] a = new int[10]; compile time error

This would create an error because a is of type float[] so it is not possible to recast it in java this way. Java is a compiled language, not interpreted.

Question 2 [5 points]

In the following code, a breakpoint is inserted on line 16 as shown. Depict the contents of the stack and heap memories when we stop at this breakpoint. (You don't need to show actual memory locations, nor do you need to do any Ascii conversions for the strings. You should show a pointer from stack to heap as required.)

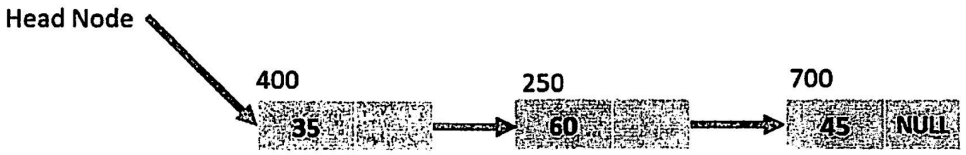
```
3  public class ClassA {
4          public static void main (String[] args){
5              boolean b0 = true;
6              String s0 = "Good";
7              method1();
8          }
9          public static void method1 (){
10             int[] a1 = {10, 20};
11             String x2 = "Day";
12             method2();
13         }
14         public static void method2(){
15             boolean b2 = true;
16             int n2 = 3;
17             int n3=20;
18         }
19 }
```



*comments (if unclear by diagram)
- each int is 4 bytes; each char is 2 bytes
- "x" would actually be an ascii code
- All arrays are contiguous in memory
- All strings are contiguous in memory

Question 3 [5 points]

For the Linked List shown below, complete the Address and Contents columns of the memory table for the head node pointer and all the nodes. (You may optionally use the Comments column if you need to). Assume that the head node pointer is stored in location **150**, and values stored in each node are **short integers that take up 2 bytes** and **all addresses take up 8 bytes** of space. The starting address of each node is indicated. NULL values may simply be indicated as 'NULL'. (Note: the addresses in the address column should preferably be starting with Head Node, then Nodes 1 - 3).



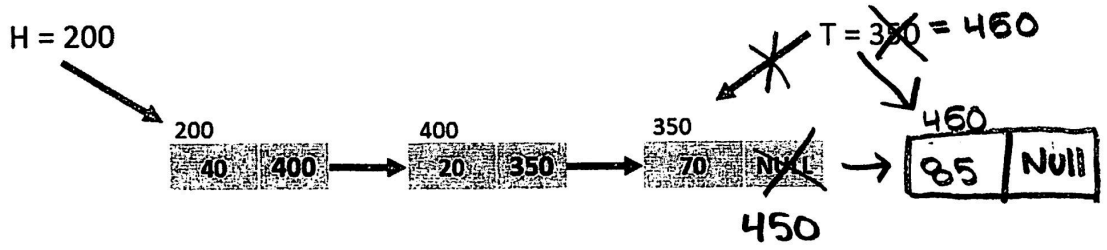
Address (From-To) (in decimal)	Contents	(Comment) (No points for comments)
150 - 157	400	Head Node
400 - 401	35	1st Node
402 - 409	250	
250 - 251	60	2nd Node
252 - 259	700	
700 - 701	45	3rd Node
702 - 709	NULL	

Question 4 [3 points]

Consider an un-sorted List of integers: List=[40, 20, 70], which is stored as a **un-sorted** Linked List data structure as shown below.

a) Show how the linked list would change after we insert a new node with content "85". Assume that this new node is stored starting at memory address 450, and that the insert algorithm uses the tail pointer (T) for insertion at the end.

b) What would be the best case, worst case, and average time-complexity in Big-O notation for searching for a key in such a list with N elements?



Best case: O(1)
Average: O(N)
Worst case: O(N)

Question 5 [6 points]

a) Give one advantage and one disadvantage of a recursive algorithm as compared to an iterative algorithm.

Advantage:
- can be more readable / understandable (easier to write)

Disadvantage:
- can use more space on the stack

b) Give one advantage and one disadvantage of a Linked List over a (static) array.

Advantage:
- more efficient with storage if number of elements is unknown at the start (↳ can't change size)

Disadvantage:
- In the case where you know the exact number of elements, array is more efficient w/ storage (use extra space for address)

c) Briefly describe the difference between static and non-static methods and variables.

Static: part of the class, do not need to instantiate an instance of the class to access.

Nonstatic: part of an instance of a class. must instantiate an instance of the class to access.

Question 6 [6 points]

Consider the following 2 methods. Assume that the length of input array **a** is 'n' and the contents are random integers between 0 and 100.

What are the Best- and Worst-case time complexities of these methods using the Big-O notation? Briefly explain your reasoning.

(a)

```
static int method1 (int[] a) {
    int sum = 0;
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < 10; j++) {
            sum = sum + a[i] + i + j;
        }
    }
    return sum;
}
```

Best: O(n)

Worst: O(n)

Best/worst are the same because the double loop always runs through completely.
the 1st loop depends on N; the 2nd loop does not ⇒ O(n) because it is a linear increase

(b)

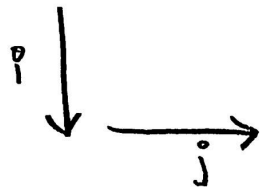
```
static int method2(int[] a) {
    int sum = 0;
    for (int i=0; i < a.length; i++) {
        for (int j=0; j < a.length; j++) {
            if (a[i] > 10)
                sum = sum + a[i] + j;
            else return sum;
        }
    }
    return sum;
}
```

Best: O(1)

→ this is the case where a[0] <= 10 which will return and exit, regardless of the size of N.

Worst: $O(n^2)$

This is due to the double for loop traversing 2 dimensions. i & j both increase by n. else condition is never satisfied.



[BLANK PAGE – FOR ROUGH WORK]

1101 1111

1 + 2 + 4 + 8 + 16 + 64 + 128

10 + 5 + 80 + 128

95 + 128

128
95
223

16 • 13 + 15

116
•13
48
160
208
+15
223

208
+11
219

0: 0 - 3
1: 4 - 7
2: 8 - 11
3: 12 - 15
4: 16 - 19
5: 20 - 23
6: 24 - 27
7: 28 - 31