Exam 1

Graded

Student

Adam Fenjiro

Total Points

66.75 / 85 pts

Question 1

1 — **18.5** / 20 pts

1.1 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.2 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.3 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.4 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.5 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.6 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.7 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.8 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.9 (no title) — **2** / 2 pts

✔ **– 0 pts** Correct

1.10 (no title) — **0.5** / 2 pts

✔ **– 1.5 pts** First 3 correct

Question 2

(no title) **10** / 10 pts

2.1 **a)** **3** / 3 pts

✔ **– 0 pts** Correct

2.2 **b)** **3** / 3 pts

✔ **– 0 pts** Correct

2.3 **c)** **4** / 4 pts

✔ **– 0 pts** Correct

Question 3

(no title) **9** / 10 pts

3.1 **a)** **3** / 4 pts

✔ **– 1 pt** not pseudo code

3.2 **b)** **3** / 3 pts

✔ **– 0 pts** Correct

3.3 **c)** **3** / 3 pts

✔ **– 0 pts** Correct

Question 4

(no title) **15.5** / 25 pts

4.1 **4.1 a)** **1** / 1 pt

✔ **– 0 pts** Correct

4.2 **4.1 b)** **1** / 1 pt

✔ **– 0 pts** Correct

4.3 **4.1 c)** **1** / 2 pts

✔ **– 1 pt** correct lower bound is Omega(n)

4.4 **4.2 a)** **1** / 1 pt

✔ **– 0 pts** Correct

4.5 **4.2 b)** **1** / 1 pt

✔ **– 0 pts** Correct

4.6 **4.2 c)** **1** / 2 pts

✔ **– 1 pt** lower bound should be Omega(n)

4.7 **4.3 part1** **4** / 4 pts

✔ **– 0 pts** Correct

4.8 **4.3 worst case** **0** / 1.5 pts

✔ **– 1.5 pts** wrong answer. Should be O(logn)

4.9 **4.3 best case** **0** / 1.5 pts

✔ **– 1.5 pts** wrong answer. Should be Omega(logn)

4.10 **a) i** **1** / 1 pt

✔ **– 0 pts** Correct

4.11 **a) ii** **0** / 1 pt

✔ **– 1 pt** Wrong answer. should be 0

4.12 **b) i** **1** / 1 pt

✔ **– 0 pts** Correct

4.13 **b) ii** **0** / 1 pt

✔ **– 1 pt** wrong answer. Should be 10

4.14 **c) i** **0** / 1.5 pts

✔ **– 1.5 pts** Wrong answer. SHould be -1. 4. -1, 3, -1

4.15 **c) ii** **0** / 1 pt

✔ **– 1 pt** wrong answer. Should be 6

4.16 **best case growth rate** **1.5** / 1.5 pts

✔ **– 0 pts** Correct

4.17 **worst case growth rate** **2** / 2 pts

✔ **– 0 pts** Correct

Question 5

(no title) **5.75** / 10 pts

5.1 **a)** **2** / 2 pts

✔ **– 0 pts** Correct

5.2 **b)** **0.5** / 1.5 pts

✔ **– 1 pt** wrong index order

5.3 **c)** **1.5** / 1.5 pts

✔ **– 0 pts** Correct

5.4 **d)** **0** / 1 pt

✔ **– 1 pt** wrong answer

5.5 **e)** **0** / 1 pt

✔ **– 1 pt** wrong answer

5.6 **f)** **1.5** / 1.5 pts

✔ **– 0 pts** Correct

5.7 **g)** **0.25** / 1.5 pts

✔ **– 1.25 pts** not close bound

Question 6

Tree **8** / 10 pts

✔ **– 0 pts** Correct

✔ **– 2 pts** base case is wrong. One node should return 0. Your code returns 1

Data Structures First Exam, Spring 2023

(85 points)

Please write your name and user name clearly. Account name/User name is the part before @ in your email address. It is not your M number. For example, my email is ruihong@mtu.edu and my username is ***ruihong***

Your Name: Adam FENJIRO **Account Name/User Name :** afenjiro

Formula:

$$1 + 2 + 3 + \cdots + (n-1) + n = \frac{n * (n+1)}{2}$$

$$1 + 2 + 3 + \cdots + (n-1) = \frac{(n-1) * n}{2}$$

The sequence: 1, 2, 4, 8, 16, ... 2^k is same as the sequence 2^0, 2^1, 2^2, 2^3, 2^4, ... 2^k. Let's say the last number is n. We have $n=2^K$, then $k = \log_2 n$.

Q2:

c. The head should be the front of the list.
This is because the head represents the first added element, and when dequeuing from the queue, we remove the element at the head (which was the oldest inserted element. This respects the FIFO principle of a queue.

1. (20 points) Multiple choices

1) Which data structure has a fixed size, Linked List or Array? Array

2) Which data structure is good at random access, Linked List or Array? Array

3) Which data structure is good at inserting and removing data in the middle, Linked List or Array? LinkedList

4) *What does ADT stand for?* Abstract Data Structure

5) Which abstract data structure has the property of First In First Out: Queue

6) Which abstract data structure has the property of First In Last Out: Stack

7) What are the two main methods for Stack ADT<E>? Include the arguments and return type

   a. push(E)

   b. E pop()

8) What are the two main methods for Queue ADT<E>? Include the arguments and return type

   a. enqueue(E)

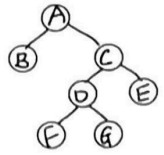
   b. E dequeue()

9) A binary tree of height 1 has at most 3 nodes, like this

What is the maximum number of nodes for a binary tree of height 3? 15

What is the maximum number of nodes for a binary tree of height 4? 31

10) We studied in class that a binary tree could be represented using array.
Show the array to represent the following binary tree.



A	B	C	D	E	F	G												

2. (10 points) Questions about queue implementation.

a) Let's use array to implement Queue. We use **index 0 as the front of the queue**. Given array size 5, perform the following operations. Show the data in the array. Don't show the data that has been dequeued.

enqueue('A'), dequeue(), enqueue('B'), dequeue(), enqueue('C'), dequeue(), enqueue('D')
Show the **final data** here. Use scratch paper to work on the intermediate steps.

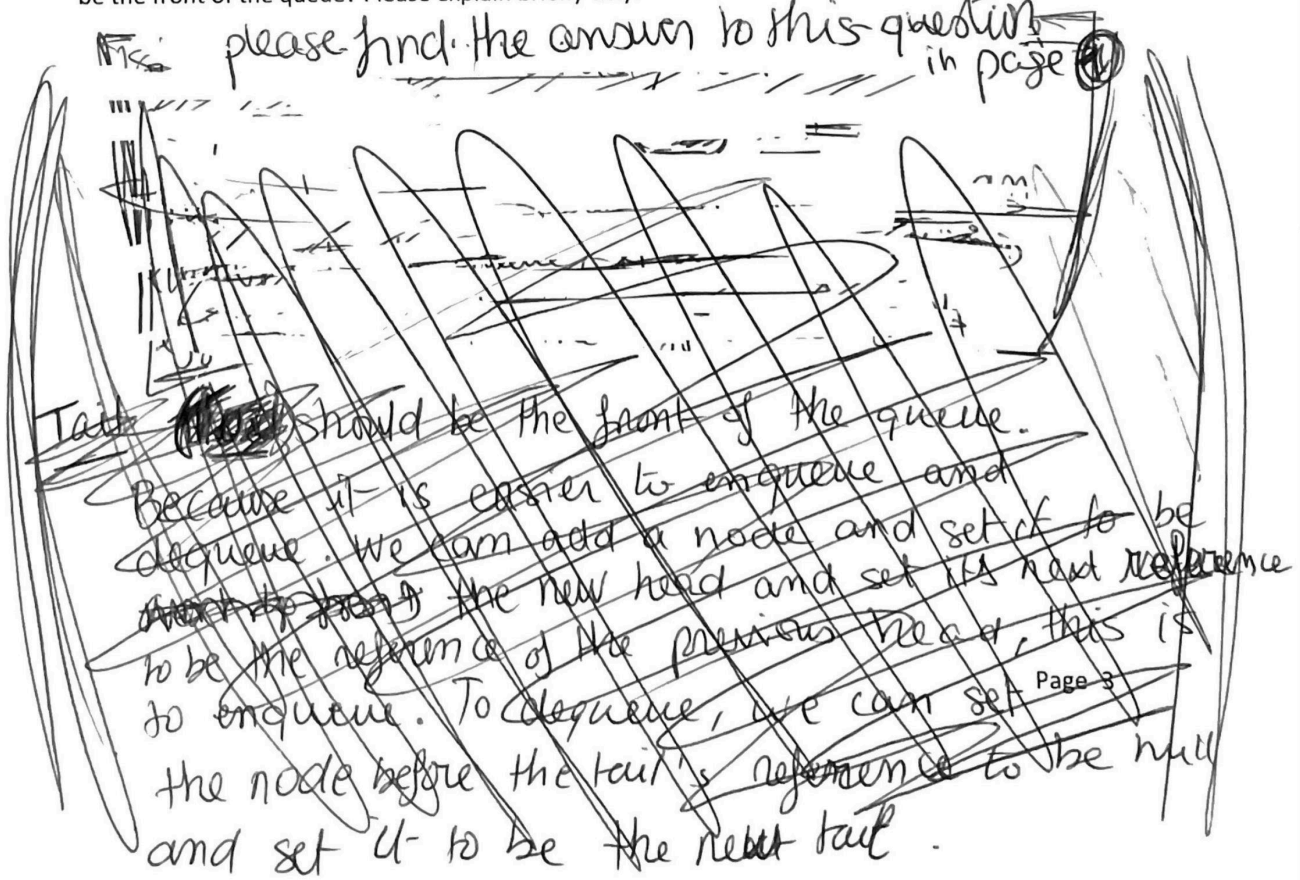
D				

b) Given array size 5, after performing the following operations using **circular array implementation** of queue, show the data in the array. Don't show the data that has been dequeued.

enqueue('A'), dequeue(), enqueue('B'), dequeue(), enqueue('C'), dequeue(), enqueue('D')
Show the **final data** here. Use scratch paper to work on the intermediate steps.

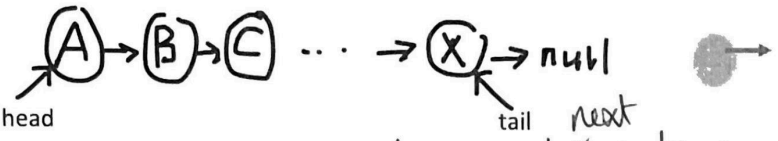
			D	

c) Let's use singly linked list to implement Queue. Should **head** be the front of queue or should **tail** be the front of the queue? Please explain briefly why.



3. (10 points) Linked List, Array and Stack.

a) **Write two lines of pseudo code** to add a node called **v** to the end of a singly linked list as below. Each node has a pointer called **next** to point to the next node.



n = new node which pointer is null
set tail.next to n
set tail to n

b) Let's use array to implement Stack. Should **index 0** be the top of the stack or the bottom of the stack? Please explain briefly why.

index 0 should be at the bottom of the stack, it is more efficient. Adding element can be done in constant time while adding to the top requires shifting all existing elements.

c) Given array size 5, show the data in array after the following operation with your design of stack in b): push ('A'), push('B'), pop(), push('C'), push('D'), pop(), push('E')

A	C	E		

4. (25 points) Analyze the complexity for the following algorithms. Describe the worst case and best case running time of the following pseudocode functions in Big-Oh notation for upper bound and Big-Omega for the lower bound from the lists below. You should choose the time complexity with the closest bound.

Upper bound: O(1), O(log n), O(n), O(nlogn), $O(n^2)$, $O(n^3)$, $O(2^n)$
Lower bound: Ω(1), Ω(log n), Ω(n), Ω(nlogn), $\Omega(n^2)$, $\Omega(n^3)$, $\Omega(2^n)$

1) (4 points) The following algorithm calculates the value of 1*2*3*…*n

Line#	**Algorithm** fac (n) **Input**: positive integer n>=1. **Output**: The product of all the integers 1, 2, ... n
1	product ← 1
2	**for** i from 1 to n **do**
3	product ← product * i
4	return product

a. What does fac(3) return? 6

b. How many times does the line 3 get executed **in terms of n after calling fac(n)?**

n

c. Running time complexity in terms of n: O(n), Ω(1)

2) (4 points) The following algorithm calculates the value of 1*2*3*…*n

Line#	**Algorithm** fac2 (n) **Input**: positive integer n>=1. Global variable count with value 0 **Output**: The product of all the integers 1, 2, ... n
1	count ← count +1;
2	if n=1 return 1
3	product ← n * fac2 (n-1)
4	return product

a. What does fac2(3) return? 6

b. What is the value of the global variable `count` in terms of n after calling fac2(n)?

n

c. Running time complexity in term of n : O(n), Ω(1)

3) (7 points) Study the function below.

```
   Algorithm fun1(A, n)
      Input: array A of size n, that is A[0..n-1]
      Output: Some value in A will be changed.

1     i ← 1
2     while i < n do
3        A[i] ← A[i] + i
4        i ← i * 2
```

If A=[1, 1, 1, 1, 1, 1, 1, 1, 1, 1], after calling fun1(A, 10), show the elements in A

A=

1	2	3	1	5	1	1	1	9	1

Running time of the algorithm fun1() in terms of n: $O(n\log(n))$, $\Omega(n)$

4) (10 points) The following algorithm find the next greater element for every element in an array.

The next greater element of some element x in array is the **first greater** element that is **to the right** of x in the same array. If there is no element greater, then the next greater element will be set as -1.

```
   Algorithm nextGreaterElement(A, B, n)
      Input: Array A and B have size n. Array A has n data items.
      Output: Element in Array B is the next greater element of Array A or -1
   if there is no greater element.

1     for i from 0 to n - 1 do
2        j ← i+1
3        B[i] ← -1

4        while j < n and A[j] < A[i] do
5           j++

6        if j < n then
7           B[i] ← A[j]

8     return count
```

a. If A=[1, 2, 3, 4, 5], after calling `nextGreaterElement(A, B, 5)`,

   i. What will be the data in B: [2, 3, 4, 5, -1]

   ii. How many times was line 5 executed? 4

b. If A=[5, 4, 3, 2, 1], after calling `nextGreaterElement(A, B, 5)`,

   i. What will be the data in B: [-1, -1, -1, -1, -1]

   ii. How many times was line 5 executed? 10

c. If A=[5, 1, 4, 2, 3], after calling `nextGreaterElement(A, B, 5)`,

   i. What will be the data in B: [4, 3, -1, -1, -1]

   ii. How many times was line 5 executed? 9

d. Best case time complexity: $\Omega(n)$

e. Worst case time complexity: $O(n^2)$

5. (10 points) Stack Application

Stack is a very useful data structure. It could be used to solve the next greater element problem that you saw earlier in the exam. Below is the java code using java.util.Stack. Read the comment and the code carefully. Then answer the questions after the code.

```
public static void nextGreaterElement(int[] A, int[] B, int n) {

      // We will push the index for the element that we need to find the next greater
element for as we go
      Stack<Integer> S = new Stack<Integer>();

      // We need to find the next greater element for the first element,
      // so, let's push the index 0 into the stack
      S.push(0);

      // Loop through all the elements in A starting from index 1
      // and check if it is the next greater element for some previous element(s)
      for (int i=1; i<n; i++) {

         // If A[i] is greater than the element whose index is on the top of the stack,
         // then A[i] is the next greater element for that element
         while (S.size()>0 && A[i] > A[S.peek()]) {

            int j=S.pop();

            // A[i] is the next greater element for A[j]
            B[j]=A[i];

            // Print the information
            System.out.println("B[" + j + "]=" + A[i]);
         }

         // We don't know what is the next greater element for A[i],
         // so let's push the index i into the stack.
         S.push(i);
      }

      // If there are still some indexes in the stack,
      // it means there is no greater element for them.
      // Therefor set the next greater element to be -1
      while (S.size()>0) {
         int j=S.pop();
         B[j]=-1;

         // Print the information
         System.out.println("B[" + j + "]=" + "-1");
      }
}
```

a. If A=[1, 2, 3, 4, 5], when we call `nextGreaterElement(A, B, 5)`, the program will print 5 lines. Please fill the blank of the output. The order has to match.

B[0] = 2
B[1] = 3
B[2] = 4
B[3] = 5
B[4] = -1

b. If A=[5, 4, 3, 2, 1], when we call `nextGreaterElement(A, B, 5)`, the program will print 5 lines. Please fill the blank of the output. The order has to match.

B[0] = -1
B[1] = -1
B[2] = -1
B[3] = -1
B[4] = -1

c. If A=[5, 1, 4, 2, 3], when we call `nextGreaterElement(A, B, 5)`, the program will print 5 lines. Please fill the blank of the output. The order has to match.

B[1] = 4
B[3] = 3
B[4] = -1
B[2] = -1
B[0] = -1

d. When we call `nextGreaterElement(A, B, n)`, how many times is S.push() called in terms of n? 4

e. When we call `nextGreaterElement(A, B, n)`, how many times is S.pop() called in terms of n? 2

f. Best case time complexity: Ω(N)

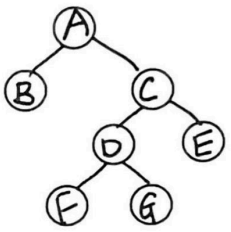
g. Worst case time complexity: O(N^2)

6. (10p) Tree

Implement the function height(v) to find out the height of a subtree rooted at node v.

For example:

- The height of the subtree rooted at node C is 2.
- The height of the subtree rooted at node B is 0.
- The height of the subtree rooted at node A is 3



Implementation Requirement:

- Use **recursive** functions calls
- You may **only** use the following methods: **v.left() and v.right()**

Algorithm height(v)
Input: a node v
Output: The height of the subtree rooted at node v

```
public int height(v){
    if (v == null){
    return 0;
    }
    else{
    int leftHeight = height(v.left());
    int rightHeight = height(v.right());
    if(leftHeight < rightHeight){
        return rightHeight + 1;
    }
    else{return leftHeight + 1;}
    }
}
```

// Base case

// recursive

// printing height