Part A: Bounds

For each question in this section, give the unqualified big-O, big-Ω, and big-Θ bounds for the specified function. If the big-Θ bound does not exist, write **DNE**. For this section you are not required to show any work or give a proof unless stated otherwise.

Question 1 [**3 points**]

$$f_1(n) = \sum_{i=1}^{n^3} (20n^2 + 5i)$$

Answer

Variant A: n^6 for all
Variant B: n^4 for all
Variant C: n^6 for all
Variant D: n^4 for all

Point Breakdown

- **(+1 pt)** For correct O
- **(+1 pt)** For correct Omega
- **(+1 pt)** For a consistent Theta

Question 2 [**3 points**]

$$f_2(n) = \sum_{i=0}^{\log(n^5)} \sum_{j=0}^{i-1} 2^j$$

Answer

Variant A: n^5 for all
Variant B: n^3 for all
Variant C: n^2 for all
Variant D: n^6 for all

Point Breakdown

- **(+1 pt)** For correct O
- **(+1 pt)** For correct Omega
- **(+1 pt)** For a consistent Theta

Question 3 [**3 points**]

$$f_3(n) = \begin{cases} \log(n) + 9n\log(n) + n^2 & \text{if n is odd} \\ n\log(n) + 5\log(n) & \text{if n is even} \end{cases}$$

Answer

Variant A: $O(n^2)$, $\Omega(n\log(n))$, Θ DNE
Variant B: $O(n^4)$, $\Omega(n\log(n))$, Θ DNE
Variant C: $O(n\log(n))$, $\Omega(n\log(n))$, $\Theta(n\log(n))$
Variant D: $O(n^3)$, $\Omega(n)$, Θ DNE

Point Breakdown

- **(+1 pt)** For correct O
- **(+1 pt)** For correct Omega
- **(+1 pt)** For a consistent Theta

Question 4 [**6 points**]

Prove that the following function is in $O(n^4)$:

$$f_4(n) = 39n^2 + 10 + 5n^4$$

Answer

Break into terms
$39n \leq c \cdot n^4$ is true when $c = 39$, $n \geq 0$
$10 \leq c \cdot n^4$ is true when $c = 10$, $n \geq 1$
$5n^4 \leq c \cdot n^4$ is true when $c = 5$, $n \geq 0$
Therefore if $c = 54$ and $n \geq 1$, $39n^2 + 10 + 5n^4 \leq c \cdot n^4$ which means $f_4 \in O(n^4)$

Point Breakdown

- **(+1 pt)** For including each term in the proof in some way
- **(+1 pt)** For a valid choice for c
- **(+1 pt)** For a valid choice for n_0
- **(+1 pt)** For at some point having the definition of big-O

Part B: Hash Table Concepts

Question 1 [**12 points**]

Consider a Hash Table that uses chaining to resolve collisions, but instead of each bucket being a LinkedList, each bucket is a Sorted ArrayList. State the expected and tight unqualified runtime for each Hash Table operation discussed in class.

`insert(T elem)`	`contains(T elem)`	`remove(T elem)`
Expected:	Expected:	Expected:
Unqualified:	Unqualified:	Unqualified:

Answer

Expected: O(1), O(1), O(1)
Unqualified: O(n), O(log(n)), O(n)

Point Breakdown

- **(+2 pt)** per correct answer

Question 2 [**8 points**]

In class we described an implementation of the Set ADT using a Hash Table. In at most a few sentences, describe what concrete changes we would need to make in order to implement the Bag ADT using a Hash Table. As a reminder, the Bag ADT is unordered (like a Set), but allows for duplicate elements.

Answer

Any of the following would be a valid answer:
- Instead of just storing the element, store an object holding the element and a count (like we did for PA1)
- Treat it like a map, where the keys are the elements and the values are the counts
- When using chaining, just allow multiples of the same element in each bucket
- If using open addressing, allow for multiple of the same value to be inserted

Point Breakdown

- **(8 pt)** For a correct answer matching one of the above

Part C: Hash Table Implementation

For all questions in this section, you may assume that $hash_1$ and $hash_2$ are hash functions that return the following values for keys A through F:

	A	B	C	D	E	F
$hash_1$	37	13	93	24	64	58
$hash_2$	60	86	27	81	38	57

Question 1 [**6 points**]

Consider a Hash Table that has 10 buckets, uses $hash_1$ as its hash function, and resolves collisions using **chaining**. You may assume no rehash is required. State which bucket each element will end up in if elements A through F are inserted in alphabetical order.

A: ____ B: ____ C: ____ D: ____ E: ____ F: ____

Answer

Variant A: 7, 3, 3, 4, 4, 8
Variant B: 0, 4, 4, 1, 1, 5
Variant C: 1, 5, 5, 6, 6, 0
Variant D: 0, 9, 9, 7, 7, 1

Point Breakdown

- **(+1 pt)** For each element in the correct bucket

Question 2 [**6 points**]

Consider a Hash Table that has 10 buckets, uses $hash_1$ as its hash function, and resolves collisions using **open addressing**. You may assume no rehash is required. State which bucket each element will end up in if elements A through F are inserted in alphabetical order.

A: ____ B: ____ C: ____ D: ____ E: ____ F: ____

Answer

Variant A: 7, 3, 4, 5, 6, 8
Variant B: 0, 4, 5, 1, 2, 6
Variant C: 1, 5, 6, 7, 8, 0
Variant D: 0, 9, 1, 7, 8, 2

Point Breakdown

- **(+1 pt)** For each element in the correct bucket

Question 3 [**6 points**]

Consider a Hash Table that has 10 buckets, uses $hash_1$ and $hash_2$ as its hash functions, and resolves collisions using **cuckoo hashing** exactly as you did in PA3. You may assume no rehash is required. State which bucket each element will end up in if elements A through F are inserted in alphabetical order.

A: ____ B: ____ C: ____ D: ____ E: ____ F: ____

Answer

Variant A: 7, 6, 3, 1, 4, 8
Variant B: 0, 8, 4, 3, 1, 5
Variant C: 1, 3, 5, 8, 6, 0
Variant D: 0, 4, 9, 6, 7, 1

Point Breakdown

- **(+1 pt)** For each element in the correct bucket (based on their number of buckets)

Question 4 [**2 points**]

If the Hash Table with **chaining** described in Question 1 was given a maximum load factor of 0.3 which element would trigger a rehash upon being inserted? If no element would trigger a rehash write "None".

Answer

Variant A: D
Variant B: E
Variant C: F
Variant D: E

Point Breakdown

- **(2 pt)** For the correct answer

Part D: Code Runtime

This part of the exam pertains to the following function and labeled boxes. For several questions, you will be asked to provide summations of runtimes labeled with line numbers. See below for an example of a summation for lines 11-13.

```
1  public Integer MyFunction(List<RecordA> recordsN,
2                            List<RecordB> recordsM)
3  {
4    Accumulator accumulator =
5      InitAccumulator(recordsN.size(), recordsM.size());
6
7    for(i : recordsN) {
8      InsertIntoAccumulator(accumulator, i);
9    }                                          A (lines 7-9)
                                                B (lines 4-9)
10
11   CleanupAccumulator(accumulator);
12
13   Integer result = 0;
14
15   for(j : recordsM) {
16
17     if( IsValid(j) ) {
18       result += RetrieveFromAccumulator(accumulator, j);
19     } else {
20       result += 1;
21     }                                        C (lines 17-21)
22
23   }                                          D (lines 15-23)
24
25   return result;                             E (lines 11-25)
26 }
```

Example

$$\underbrace{O(1)}_{\texttt{Line 11}} + \underbrace{O(1)}_{\texttt{Line 13}}$$

The table below presents *tight* bounds on the runtime of the functions called by **MyFunction**. The letters N and M indicate the exact values of `recordsN.size()` and `recordsM.size()` respectively. The expression $|A|$ means the number of times that `InsertIntoAccumulator` has *previously* been called on A (i.e., the size of A). You may assume that iterating over a list is constant-time per element. All bounds are unqualified.

Function	Big-O	Big-Ω
`InitAccumulator(N, M)`	$O(N+M)$	$\Omega(N)$
`InsertIntoAccumulator(A, i)`	$O(\|A\|)$	$\Omega(\|A\|)$
`CleanupAccumulator(A, i)`	$O(1)$	$\Omega(1)$
`IsValid(j)`	$O(1)$	$\Omega(1)$
`RetrieveFromAccumulator(A, i)`	$O(\|A\|^2)$	$\Omega(\|A\|)$

Variant A

Big-O

Block A: $\underbrace{\sum_{X:N}}_{\text{line 7}} \underbrace{O(x)}_{\text{line 8}} = O(N^2)$

Block B: $\underbrace{O(N+M)}_{\text{line 5}} + [\text{Block A}] = O(N^2+M)$

Block C: $\underbrace{O(1)}_{\text{line 17}} + \underbrace{\{\ 1\ \}}_{\text{lines 17-21}} = O(N^2)$

Block D: $\underbrace{\sum_{X:M}}_{\text{line 15}} [\text{Block C}] = O(M \cdot N^2)$

Block E: $\underbrace{O(1)}_{\text{line 11}} + \underbrace{O(1)}_{\text{line 13}} + [\text{Block D}] + \underbrace{O(1)}_{\text{line 25}} = O(M \cdot N^2)$

Total: $O(M \cdot 2^N)$

Big Ω

$\underbrace{\sum_{X:N}}_{\text{line 7}} \underbrace{\Omega(x)}_{\text{line 8}} = \Omega(N^2)$

$\underbrace{\Omega(N)}_{\text{line 5}} + [\text{Block A}] = \Omega(N^2)$

$\underbrace{\Omega(1)}_{\text{line 17}} + \underbrace{\{\ |A| = N\ \}}_{\text{lines 17-21}} = \Omega(1)$

$\underbrace{\sum_{X:M}}_{\text{line 15}} [\text{Block C}] = \Omega(M)$

$\underbrace{\Omega(1)}_{\text{line 11}} + \underbrace{\Omega(1)}_{\text{line 13}} + [\text{Block D}] + \underbrace{\Omega(1)}_{\text{line 25}} = \Omega(M)$

$\Omega(N^2+M)$

Variant B

Big-O

Block A: $\underbrace{\sum_{X:N}}_{\text{line 7}} \underbrace{O(2^X)}_{\text{line 8}} = O(2^N)$

Block B: $\underbrace{O(N)}_{\text{line 5}} + [\text{Block A}] = O(2^N)$

Block C: $\underbrace{O(1)}_{\text{line 17}} + \underbrace{\{\ \log(|A|) = \log(N)\ \}}_{\text{lines 17-21}} = O(\log(N))$

Block D: $\underbrace{\sum_{X:M}}_{\text{line 15}} [\text{Block C}] = O(M \log(N))$

Block E: $\underbrace{O(N^2)}_{\text{line 11}} + \underbrace{O(1)}_{\text{line 13}} + [\text{Block D}] + \underbrace{O(1)}_{\text{line 25}} = O(N^2 + M\log(N))$

Total: $O(2^N + M\log(N))$

Big Ω

$\underbrace{\sum_{X:N}}_{\text{line 7}} \underbrace{\Omega(1)}_{\text{line 8}} = \Omega(N)$

$\underbrace{\Omega(1)}_{\text{line 5}} + [\text{Block A}] = \Omega(N)$

$\underbrace{\Omega(1)}_{\text{line 17}} + \underbrace{\{\ \log_2(|A|) = \log(N)\ \}}_{\text{lines 17-21}} = \Omega(1)$

$\underbrace{\sum_{X:M}}_{\text{line 15}} [\text{Block C}] = \Omega(M)$

$\underbrace{\Omega(N)}_{\text{line 11}} + \underbrace{\Omega(1)}_{\text{line 13}} + [\text{Block D}] + \underbrace{\Omega(1)}_{\text{line 25}} = \Omega(N+M)$

$\Omega(N+M)$

Variant C

Big-O

Block A: $\underbrace{\sum_{X:N}}_{\text{line 7}} \underbrace{O(2^X)}_{\text{line 8}} = O(2^N)$

Block B: $\underbrace{O(M)}_{\text{line 5}} + [\text{Block A}] = O(M+2^N)$

Block C: $\underbrace{O(1)}_{\text{line 17}} + \underbrace{\{\ |A| = N\ \}}_{\text{lines 17-21}} = O(N)$

Block D: $\underbrace{\sum_{X:M}}_{\text{line 15}} [\text{Block C}] = O(M \cdot N)$

Block E: $\underbrace{O(N)}_{\text{line 11}} + \underbrace{O(1)}_{\text{line 13}} + [\text{Block D}] + \underbrace{O(1)}_{\text{line 25}} = O(M \cdot N)$

Total: $O(M \cdot N + 2^N)$

Big Ω

$\underbrace{\sum_{X:N}}_{\text{line 7}} \underbrace{\Omega(1)}_{\text{line 8}} = \Omega(N)$

$\underbrace{\Omega(M)}_{\text{line 5}} + [\text{Block A}] = \Omega(M+N)$

$\underbrace{\Omega(1)}_{\text{line 17}} + \underbrace{\{\ \log_2(|A|) = \log N\ \}}_{\text{lines 17-21}} = \Omega(1)$

$\underbrace{\sum_{X:M}}_{\text{line 15}} [\text{Block C}] = \Omega(M)$

$\underbrace{\Omega(1)}_{\text{line 11}} + \underbrace{\Omega(1)}_{\text{line 13}} + [\text{Block D}] + \underbrace{\Omega(1)}_{\text{line 25}} = \Omega(M)$

$\Omega(M+N)$

Variant D

Big-O

Block A: $\sum_{x=0}^{N} O(x) = O(N^2)$ — line 7, line 8

Block B: $O(M) + [\text{Block A}] = O(M+N^2)$ — line 5

Block C: $O(1) + \{2^{|A|} = 2^N\} = O(2^N)$ — line 17, lines 17–21

Block D: $\sum_{x=M}^{} [\text{Block C}] = O(M 2^N)$ — line 15

Block E: $O(2^M) + O(1) + [\text{Block D}] + \Omega(1) = O(M \cdot 2^N)$ — line 11, line 13, line 25

Total = $O(M \cdot 2^N)$

Big Ω

$\sum_{x=0}^{N} \Omega(1) = \Omega(N)$ — line 7, line 8

$\Omega(1) + [\text{Block A}] = \Omega(N)$ — line 5

$\Omega(1) + \{|A|^2 = N^2\} = \Omega(1)$ — line 17, lines 17–21

$\sum_{x=M}^{} [\text{Block C}] = \Omega(M)$ — line 15

$\Omega(M) + \Omega(1) + [\text{Block D}] + \Omega(1) = \Omega(M+N)$ — line 11, line 13, line 25

$\Omega(M+N)$

Question 1 [**5 points**]

For the blocks of code labeled A and B, respectively, do each of the following:

1. Provide a tight upper (Big-O) bound on the runtime of the code in the rectangle as a **summation**.
2. Label the components of your summation that correspond to lines (i) 5, (ii) 7-9, and (iii) 8.
3. Expand the summation and simplify the resulting bound.

Answer

see above

Point Breakdown

- **(+1 pt)** The summation provided for Block A is correct.
- **(+1 pt)** The simplified runtime for Block A is exactly correct.
- **(+1 pt)** The provided summation for Block B correctly includes the `InitAccumulator` runtime (line 5).
- **(+1 pt)** The simplified runtime for Block B is exactly correct.
- **(+1 pt)** The answer's labels for code lines are sensible.

Question 2 [**5 points**]

For the blocks of code labeled C, D, and E, respectively, do each of the following:

1. Provide a tight upper (Big-O) bound on the runtime of the code in the rectangle as a **summation**.
2. Label the components of your summation that correspond to lines (i) 11, (ii) 15-21, (iii) 17-21, (iv) 17, and (v) 18, and (vi) 20.
3. Expand the summation and simplify the resulting bound.

Answer

see above

Point Breakdown

- **(+1 pt)** The summation for Block C is provided piecewise, or otherwise indicates that there are two possible runtimes.
- **(+1 pt)** The Big-O runtime of block C correctly includes *only* the greater term in the cases block.
- **(+1 pt)** The summation provided for Block D sets up the summation (relative to the Block C answer) correctly.
- **(+1 pt)** The simplified runtime for Blocks C, D, and E are correct.
- **(+1 pt)** The answer's labels for code lines are sensible.

Question 3 [**5 points**]

For the blocks of code labeled A and B, respectively, do each of the following:

1. Provide a tight lower (Big-Ω) bound on the runtime of the code in the rectangle as a **summation**.
2. Label the components of your summation that correspond to lines (i) 5, (ii) 7-9, and (iii) 8.
3. Expand the summation and simplify the resulting bound.

Answer

see above

Point Breakdown

- **(+1 pt)** The summation provided for Block A is correct.
- **(+1 pt)** The simplified runtime for Block A is exactly correct.
- **(+1 pt)** The provided summation for Block B correctly includes the `InitAccumulator` runtime (line 5).
- **(+1 pt)** The simplified runtime for Block B is exactly correct.
- **(+1 pt)** The answer's labels for code lines are sensible.

Question 4 [**5 points**]

For the blocks of code labeled C, D, and E, respectively, do each of the following:

1. Provide a tight lower (Big-Ω) bound on the runtime of the code in the rectangle as a **summation**.
2. Label the components of your summation that correspond to lines (i) 11, (ii) 15-21, (iii) 17-21, (iv) 17, and (v) 18, and (vi) 20.
3. Expand the summation and simplify the resulting bound.

Answer

see above

Point Breakdown

- **(+1 pt)** The summation for Block C is provided piecewise, or otherwise indicates that there are two possible runtimes.
- **(+1 pt)** The Big-Ω runtime of block C correctly includes *only* the lesser term in the cases block.
- **(+1 pt)** The summation provided for Block D sets up the summation (relative to the Block C answer) correctly.
- **(+1 pt)** The simplified runtime for Blocks C, D, and E are correct.
- **(+1 pt)** The answer's labels for code lines are sensible.

Question 5 [**5 points**]

Provide the simplified, tight asymptotic upper (Big-O) and lower (Big-Ω) bounds on the runtime of `MyFunction` in terms of the sizes of its two inputs (`recordsN.size()` $= N$; `recordsM.size()` $= M$).

Answer

Var A: $O(N^2M)$, $\Omega(N^2)$
Var B: $O(N^2 + N\log(N)M)$, $\Omega(N)$
Var C: $O(MN + N^2)$, $\Omega(M + N)$
Var D: $O(MN + N^2)$, $\Omega(N)$

Point Breakdown

- **(3 pt)** For having at least one bound that is consistent with the student's answers to Q1-2 (Big-O) or Q3-4 (Big-Ω).
- **(5 pt)** For having the other bound consistent with the student's prior answers.
- **(+1 pt)** Partial credit per bound provided for simply demonstrating the need to sum up blocks B and E (but e.g., not simplifying correctly).

Question 6 [**5 points**]

Provide the simplified tight, *unqualified* simplified asymptotic upper (Big-O) bound on Block **D**, if we additionally provide you with the following two facts:

- $M \gg N$ (M is guaranteed to be much bigger than N, i.e., $N \in O(M)$)
- The *amortized* runtime of `RetrieveFromAccumulator(A, j)` is $O(|A|)$.

Answer

For each question, the amortized runtime is exactly a factor of N lower than the block D runtime in Q2.

Point Breakdown

- **(2 pt)** for any answer that is strictly lower than the student's Block D runtime in Q2.
- **(3 pt)** for any answer that is correct (as below), but not fully simplified.
- **(5 pt)** for any answer that is exactly a factor of N (Variants A, B, D) or $\log(N)$ (Variant C) than the student's Block D runtime in Q2. Answers in terms of $|A|$ are acceptable if this is how they are presented in Q2.

Part e: Data Structure Design

For each of the following questions, **circle** the abstract data type **and** the data structure that best fits the requirements of the collection described in the question.

Question 1 [**5 points**]

The streets and intersections of a city for use in a route-finding application.

ADT: List Stack Queue Priority Queue Graph Set Map

Data Structure: Array LinkedList ArrayList Ring Buffer Binary Heap Edge List Adjacency List Adjacency Matrix AVL Tree Red-Black Tree Hash Table

Answer

Var A: Graph, Adjacency List
Var B: Stack, ArrayList or LinkedList
Var C: PriorityQueue, BinaryHeap
Var D: PriorityQueue, Binary Heap

Point Breakdown

- **(+2 pt)** If one selection is correct
- **(+5 pt)** If both selections are correct

Question 2 [**5 points**]

The waiting list for a class, where students are admitted in the order they joined the waiting list.

ADT: List Stack Queue Priority Queue Graph Set Map

Data Structure: Array LinkedList ArrayList Ring Buffer Binary Heap Edge List Adjacency List Adjacency Matrix AVL Tree Red-Black Tree Hash Table

Answer

Var A: Queue, LinkedList or RingBuffer
Var B: Map, HashTable
Var C: Graph, AdjacencyList
Var D: Queue, LinkedList or RingBuffer

Point Breakdown

- **(+2 pt)** If one selection is correct
- **(+5 pt)** If both selections are correct

Question 3 [**5 points**]

The character's inventory in a game. For each item, identified by its name, you need to keep track of how many copies of the item you have.

ADT: List Stack Queue Priority Queue Graph Set Map

Data Structure: Array LinkedList ArrayList Ring Buffer Binary Heap
Edge List Adjacency List Adjacency Matrix AVL Tree
Red-Black Tree Hash Table

Answer

Var A: Map, HashTable
Var B:Var C: List, LinkedList
Var D: Graph, AdjacencyList

Point Breakdown

- **(+2 pt)** If one selection is correct
- **(+5 pt)** If both selections are correct

Question 4 [**5 points**]

A checklist of every quest the story's protagonist needs to complete to succeed. We only care about a quest until it is checked off, and so we want to be able to quickly remove tasks from the checklist. Every quest is identified by a magical reference to the quest's entry in the checklist, and may be completed in any order.

ADT: List Stack Queue Priority Queue Graph Set Map

Data Structure: Array LinkedList ArrayList Ring Buffer Binary Heap
Edge List Adjacency List Adjacency Matrix AVL Tree
Red-Black Tree Hash Table

Answer

Var A: List, LinkedList
Var B: Graph, Adjacency List
Var C: Map, HashTable
Var D: Stack, ArrayList or LinkedList

Point Breakdown

- **(+2 pt)** If one selection is correct
- **(+5 pt)** If both selections are correct

Part f: Heaps and Binary Search Trees

Question 1 [**20 points**]

For each node in the tree provided below, mark down (i) the **height** of the node, (ii) the **balance factor** of the node, (iii) whether the node satisfies the **BST ordering property**, and (iv) whether the node satisfies the **min-heap *ordering* property**. Give your answer for each node as if it were the root of the tree.

Answer

Var A:
Node 23: Height 2, BF -1, BST No, Heap Yes
Node 27: Height 1, BF -1, BST No, Heap Yes
Node 29: Height 0, BF 0, BST Yes, Heap Yes
Node 31: Height 0, BF 0, BST Yes, Heap Yes

Var B:
Node 23: Height 2, BF 1, BST No, Heap Yes
Node 27: Height 0, BF 0, BST Yes, Heap Yes
Node 29: Height 1, BF -1, BST No, Heap Yes
Node 31: Height 0, BF 0, BST Yes, Heap Yes

Var C:
Node 23: Height 2, BF -1, BST No, Heap Yes
Node 27: Height 1, BF 1, BST Yes, Heap Yes
Node 29: Height 0, BF 0, BST Yes, Heap Yes
Node 31: Height 0, BF 0, BST Yes, Heap Yes

Var D:
Node 23: Height 2, BF 1, BST Yes, Heap No
Node 19: Height 0, BF 0, BST Yes, Heap Yes
Node 29: Height 1, BF 1, BST Yes, Heap Yes
Node 31: Height 0, BF 0, BST Yes, Heap Yes

Point Breakdown

- **(+1 pt)** For each correct field
- **(+1 pt)** For each fully correct node

Part g: Class Participation [Bonus]

Question 1 [**5 points**]

Name any two of the undergraduate TAs for this course (first name only is fine).

Answer

Derek, Brendan, Doniyor, Ethan, Evan, Joy, Marian, Jordan, Chris, Ronan, Alex, Shreyas, Wonwoo, Jonathan, Milos, Eric, Isabel, Julia, Robby, Jenn, Emilie, Vipassana, Alex, Gina, Matthew, Faizaan, Vrushaali

Point Breakdown

- **(+2.5 pt)** Per correct answer (only count the first two names they wrote if they write a bunch)