



A brief review of everything we covered

Chapter 1 Abstraction and Analysis



- API (application programming interface)
- Abstraction
- Pre- and post-conditions
 - Side effects
- Signature and specification of a function
- Top-down design
 - Bottom-up implementation
- Algorithm analysis
 - $T(n)$
 - O and Θ notations
 - Linear search
 - Binary search
 - Selection sort

Chapter 2 Data Abstraction



- ADT (abstract data types)
 - operator overloading
- Card
 - rank, 1-13 or 2-14
 - suit, “cdhs”
- Polymorphism, encapsulation, inheritance
- (optional) Dataset ADT (Section 2.4.2)
- Unit testing
 - HW assignment to test rank, suit, rankName, suitName
- (optional) Rational class

Chapter 3 Data Abstraction



- Python list, a sequential collection
 - Underlying implementation: array, a contiguous memory
 - Run-time complexity of insertion, deletion, appending, random access
- Deck, a sequential collection
 - A collection of Card objects
 - What container was used?
 - Shuffle method
 - Run-time complexity?
 - HW assignment: size, random for shuffling, addTop, addBottom, addRandom
- Hand (Bridge)
 - has Card objects
 - added the comparison operations to Card class
- Python dict
 - Methods
 - Run-time efficiency of dict methods
 - Linear search
 - Binary search

Part 1

True/False and Multiple Choice Questions



- 1) Python lists are implemented using contiguous arrays. *True or False?*
- 2) Inserting into the middle of array-based implementation of the list is a $\Theta(n)$ operation. *True or False?*
- 3) Looking up an item in a Python dictionary, given key, is a $\Theta(n)$ operation. *True or False?*

Part 1

True/False and Multiple Choice Questions



4) Which of the following is ***not true*** of Python list (choose only one)?

- (a) They are implemented underneath as contiguous arrays
- (b) They allow for efficient random access
- (c) They can grow and shrink dynamically
- (d) All items in a list must be of the same type

Part 1

True/False and Multiple Choice Questions



5) What operation is not supported for Python dictionaries?

- (a)** Item ordering (sorting)
- (b)** Item insertion
- (c)** Item deletion
- (d)** Item lookup

Chapter 4 Linked structures and iterators



- More about Python memory model
 - Objects
 - shallow and deep copies
- Linked list (singly linked list)
 - Run-time complexity of methods
 - Iterators
 - Iterator class
 - Generator function
- Links vs arrays
 - Run-time complexity comparison of classes methods
- HW assignment
 - add more methods to LList class
 - add data attribute `last`
 - Reverse a list (HW 9)

Chapter 11 C++ linked structures

- class LList
 - Using dynamic memory allocation
 - destructor
 - The rest of the member functions
- HW assignment
 - Doubly linked list



Chapter 5 Stacks and Queues



- Stack ADT: LIFO
 - Implementation (in Python and C++)
 - applications
 - Balanced grouping symbols (suggested practice)
 - Postfix and prefix expressions (postfix – homework assignment)
 - Function call stack
- Queue ADT: FIFO
 - Implementation in Python and C++ (suggested practice)
 - Options for implementation: circular arrays, etc.
- suggested practice:
 - Unit testing of Stack and Queue methods
- In-class work: cab company
- HW assignment
 - add more methods to LList class
 - add data attribute `last`
- Recommended reading: simulation of a cashier at a store

Part 1

True/False and Multiple Choice Questions



6) When using a linked implementation of a queue, where should insertions be done?

- (a)** at the front (head) of the linked list
- (b)** in the middle of the list
- (c)** at the end (tail) of the list
- (d)** anywhere in the list, just keep the information of its location

Chapter 6 Recursion

- Definition
- Examples:
 - Binary search
 - Fibonacci numbers
 - bad recursive definition
 - Good, similar to iterative version
 - Factorial
 - String reversal
 - Anagrams
 - Power function
 - Mergesort
 - Tower of Hanoi
- Analyzing recursion
- In-class work
- Suggested practice



Chapter 10 C++ Dynamic Memory



- Memory model: Python and C++
 - Variables
 - objects
- Dynamic memory allocation: dynamic arrays
 - Destructor
 - Constructor
 - Copy constructor
 - Assignment operator, etc.
- class List
- Reference return types
- Dynamic memory errors
 - Memory leaks
 - Accessing invalid memory
- HW assignment: class Rational

Chapter 7 Trees

- Tree terminology
- Binary trees
 - Pre-order traversal → prefix expression
 - In-order traversal → infix expression
 - Post-order traversal → postfix expression
- BST (Binary Search Tree)
 - BST property
 - Insertion (iterative and recursive)
 - Deletion
 - Find
 - Traversal
 - Used generator function (Python)
 - Visit function (Python)
 - Run-time analysis of BST operations
- In-class work
- Suggested practice: C++ version, unit tests for Python BST



Part 2

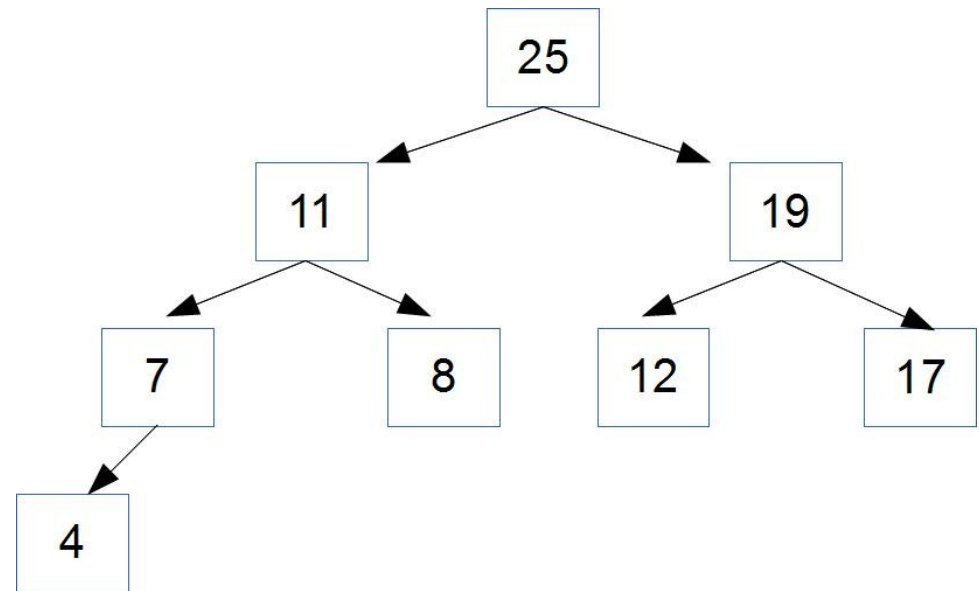


The integers: 20,12,7,14,2,5,3 and 8 are inserted into a *binary search tree*, one by one (initially the binary search tree was empty).

Draw the final tree as it appears after all the insertions, without re-balancing after each insertion operation.

Part 2

Given a binary tree, write the output of *preorder*, *inorder*, and *postorder* traversal.



Chapter 12 Templates

- Template classes
- Template functions
- vector class
- Template Stack class
- Suggested practice: template Queue class
- See other in-class work
- Homework:
 - Template average function
 - Template List class



Chapter 13 Heaps, AVL, Hash Tables



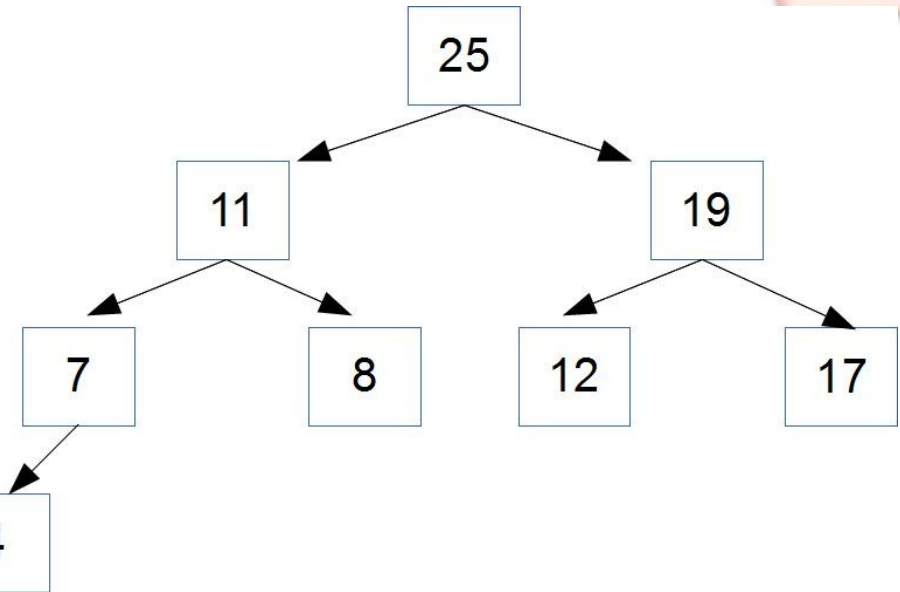
- Heap
 - heapsort
- Priority queue
 - Implementation ideas (chose to use Heap)
- AVL trees
 - AVL property
- Hash tables
 - Collision resoluion
- Suggested practice
- Homework

Part 2

Answer all questions



1) Given the following heap:



(a) Draw the underlying array representation of this heap

(b) Show the addition of the value of 10 to the heap (and the updated heap)

(c) Show the deletion of value 25 from the heap (and the updated heap)

Part 2

Answer all questions



2) The integers 20,12,7,14,2,5,3 and 8 are inserted in that order into a *priority queue*.
Give the order in which these values are retrieved.

Chapter 14 Graphs

- Graph terminology
- Representations:
 - adjacency matrix
 - adjacency list
- Shortest Path algorithms
 - BFS (Breadth First Search)
 - Dijkstra's
- Depth First Search algorithm (DFS)
- Minimum Spanning Trees
 - Kruskal's
 - Prim's



Part 2

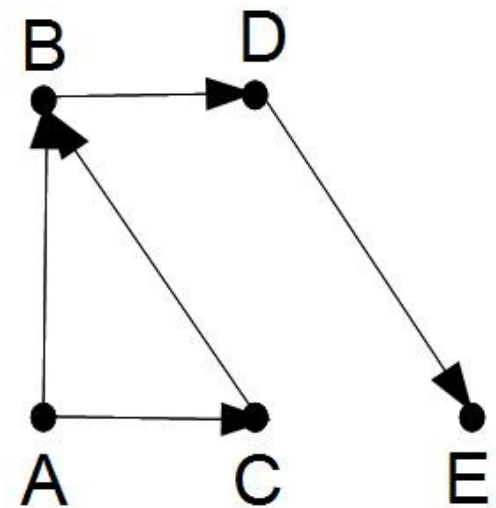
Answer all questions



4) Represent this directed, unweighted graph as a *Python list* two ways:

- as an adjacency matrix, and
- as an adjacency list.

Which representation is most efficient for a graph which has very few edges between nodes?



Part 1

True/False and Multiple Choice Questions



- 7) The *scope* of a variable refers to
- (a) the different values it can hold
 - (b) the section of code where the variable can be accessed
 - (c) the time during which memory is allocated for the variable
 - (d) the name of the variable

Part 1

True/False and Multiple Choice Questions



8) A C++ function must return a value.

True or False?

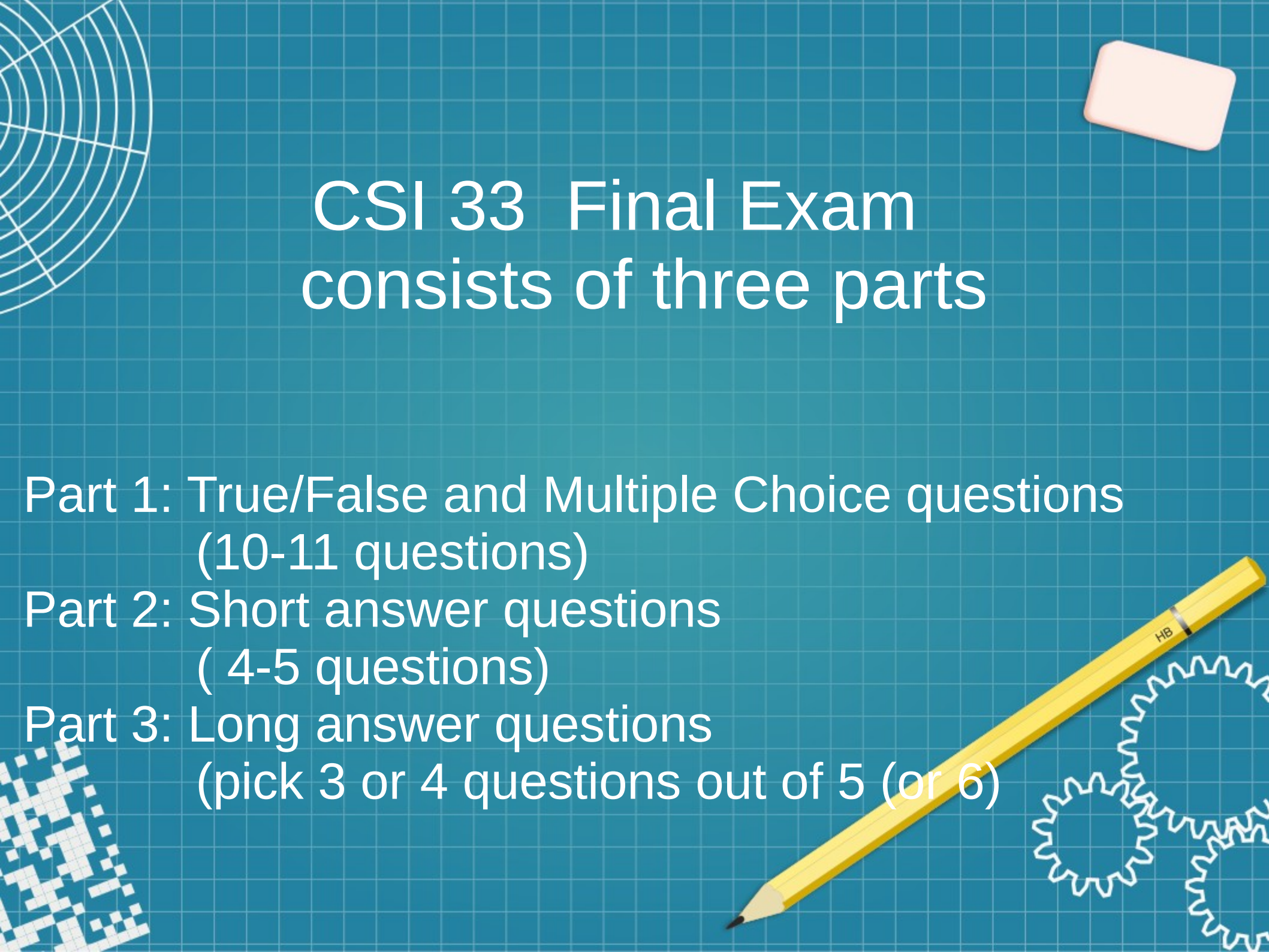
Part 3

Answer 3 out of 5 questions

1) The integers: 20,12,7,14,2,5,3 and 8 are inserted into an empty *AVL tree*, one by one.

After some of the insertion operations the tree needs to be re-balanced.

Show all the work (as the tree appears after every insertion and after every re-balancing).



CSI 33 Final Exam consists of three parts

Part 1: True/False and Multiple Choice questions
(10-11 questions)

Part 2: Short answer questions
(4-5 questions)

Part 3: Long answer questions
(pick 3 or 4 questions out of 5 (or 6))



This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. It makes use of the works of Mateus Machado Luna.

