Learning Objectives

After completing this unit, you should be able to:

- Define and give everyday examples of arrays, stacks, queues, and trees.
- Explain what a pointer is and why it's fundamental to data structures.
- Use Python lists to store and manipulate one and two-dimensional data.
- Trace the recursive implementation of a binary search tree in Python.
- Draw linked lists and binary trees that correspond to memory contents.
- Insert, delete, and find nodes (by hand) in linked lists and binary trees.
- Summarize the pros and cons of using contiguous vs linked structures.

Textbook Sections

8.1 Basic Data Structures
8.2 Related Concepts
8.3 Implementing Data Structures
8.4 A Short Case Study

Video Lectures

- Data Structures
- Binky Pointer Fun

Assignments

Act10 Data Structures; Chapter 8 Problems

Lab10 Codecademy (5 & 6); Visualizing binary trees

Unit 10 Checklist: Nov 04 – Nov 10

Before Wednesday	**Date Completed**
FINISH models 1 and 2 of Data Structures	
READ textbook 8.1 Basic Data Structures (take notes) ANSWER questions 1 and 3 in your notes	
READ textbook 8.2 Related Concepts (take notes) ANSWER questions 2 and 3 in your notes	
READ textbook 8.4 A Short Case Study (take notes) ANSWER questions 1 and 2 in your notes	
WATCH video lecture: Data Structures (take notes)	
START Lab10: Visualizing binary trees	(10 pts)
Before Friday	**Date Completed**
READ textbook 8.3 Implementing Data Structures (take notes) ANSWER questions 4 and 9 in your notes	
DO tutorial: Codecademy (5. Lists & Dictionaries)	
DO tutorial: Codecademy (6. Student Becomes the Teacher)	
START Act10 exercises (complete at least 75%)	(15 pts)
Before Monday	**Date Completed**
COMPARE your Lab10 and Act10 with the solutions in Canvas	
SUBMIT Quiz10 – 1st attempt closed: see what you don't know	
STUDY your notes, ask questions on Piazza, meet with the TAs	
SUBMIT Quiz10 – 2nd attempt open: try to get the full 10 points	(10 pts)

TAKE Exam10	(40 pts)

Activity 10: Data Structures

Model 1 Array

Consider a list of numbers:

```
lucky = [29, 16, 23, 47, 37]
```

The easiest way to store these numbers in memory is to put them side by side. For example, if the list were to begin at memory address 40:

Address:		40	41	42	43	44	45	46	47	48	49	
Value:	...	29	16	23	47	37						...
Index:												

When programming, we can access individual numbers by their ***index***. For example, the value of `lucky[1]` is `16`, and `lucky[5]` is out of range.

Questions (10 min) **Start time:** _____

1. In the diagram above, write the index below each value.

2. We call the beginning of the list the ***head*** and the end of the list the ***tail***.

   a) What is the address of the head?

   b) What is the address of the tail?

   c) What is the index of the head?

   d) What is the index of the tail?

3. What is the relationship between the index and the corresponding memory address?

4. Based on your answer to the previous question, why do indexes start at 0 instead of 1?

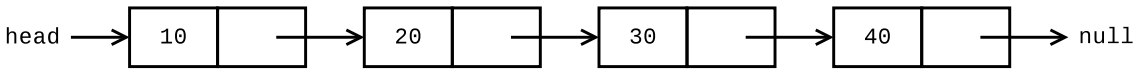
5. There are (currently) five numbers in the list. Why is `lucky[5]` out of range?

6. The statement `lucky.insert(2, 13)` changes the list to `[29, 16, 13, 23, 47, 37]`. What is the address of the head and tail after inserting 13?

7. In terms of memory operations, what does it take to insert values in the middle of an array?

Model 2 Linked List

A more flexible strategy for storing a list of numbers in memory is to use ***pointers***. A pointer is a memory address for the next item in the list.



For example, the list `[10, 20, 30, 40]` could be stored in memory this way:

Address:		74	75	76	77	78	79	80	81	82	83	
Value:	···	30	78	10	80	40	0	20	74			···
Index:												

We use the memory address 0 to represent the ***null pointer***.

Questions (10 min) **Start time:** _____

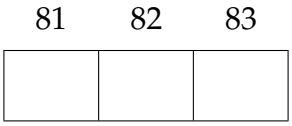
8. How many memory cells are needed for each list item?

9. In the diagram above, write the index of each list item below the corresponding memory cell. Note there are only four items in the list, so you should only have four indexes.

10. Is there a relationship between the index and the corresponding memory address? Why or why not?

11. What is the purpose of the null pointer?

12. The statement `lucky.insert(2, 25)` changes the list to `[10, 20, 25, 30, 40]`. In Model 2, change the values of memory cells `81`, `82`, and `83` to perform this insertion.



13. What memory operations does it take to insert a value in the middle of a linked list?

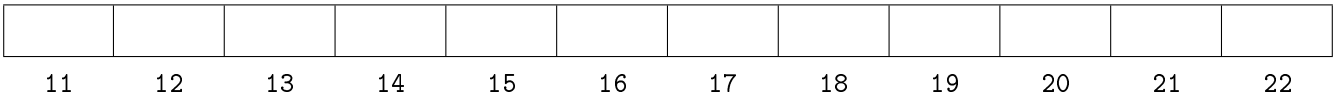
14. Summarize the pros and cons of using an array versus a linked list to represent a list of numbers.

Chapter 8: Data Abstractions

Complete the following Chapter Review Problems on pages 396–399.

#6 (inserting in contiguous array)

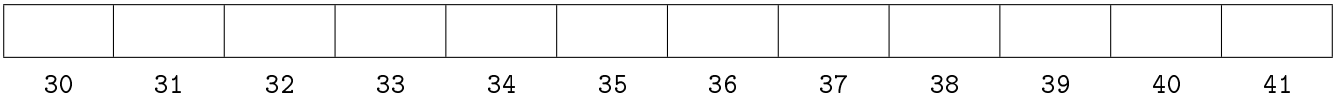
#7 (create a linked list in memory)



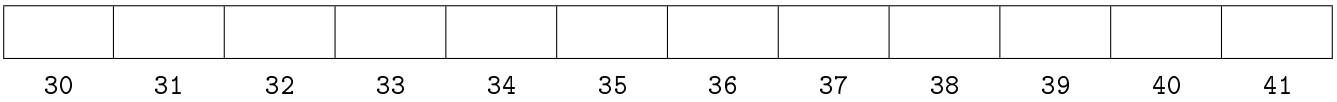
The head pointer should contain _____.

#8 (update a linked list in memory) – *circle the values that change*

After removing N:



After inserting G:

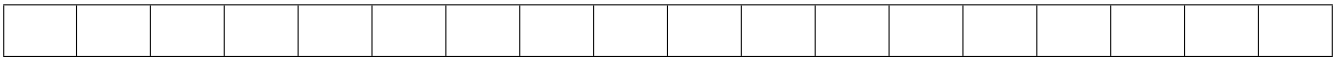


#20 (arrangements using stacks)

#25 (queue pointer arithmetic)

#29 (draw a picture of a tree)

#30 (fill in memory for a tree)



The root pointer should contain _____.

#35 (assume root is stored at 10)