**CMP4272- DATA STRUCTURES AND ALGORITHMS**

**LAB SESSION-3: Arrays, Memory Pointers, Linked Lists , Deep Copy, Shallow Copy**

**Objectives:**

* Draw box-and-pointer diagrams (BPD) for simple data structures.
* Understand the arrays and linked list data structures.
* Understand memory models for these data structures.
* Identify which data structure(s) can be used to implement which abstract data type(s).

1. **Background:**

***List ADT in Python (list)***: List type, an ordered and mutable collection of items.

We covered what a list is and what we should be able to do with it. However, we did not cover how a list is implemented. An ADT is implemented using a data structure. So how about a python list?

We can implement a list ADT using:

* + Array – Stores elements contiguously.
  + Linked Lists – Stores elements non-contiguously.

*In Python, the built-in list data type is implemented using an array-like structure. The underlying data structure is a dynamic array (array-list), which is essentially a resizable array that can grow or shrink in size as needed. The dynamic array allows Python lists to support constant-time access to elements by index, as well as dynamic resizing to accommodate varying numbers of elements.*

**Memory Models** – How variables of different types and objects are stored in memory. We used box and pointer diagrams (BPD).

**Deep Copy Vs Shallow Copy**

A shallow copy creates a new object, but it doesn't create new objects for the elements within the original object. It copies the top-level structure and references of the original object. However, the inner objects (elements) are still references to the same objects as those in the original.

A deep copy creates a new object and recursively copies all objects found in the original object. It creates fully independent copies of the original object and all objects nested within it.

1. **Getting started: Complete the following exercises before you attempt the mandatory ones provided under Lab submission Exercises in the next section.**

**Make use of the Memory Visualization tool:** [**https://werp.site/cs-toys/memory.html**](https://werp.site/cs-toys/memory.html)

**Note : Click on the settings button and set the Allocate memory to Randomly**

A screenshot of a computer

Description automatically generated

**Exercise-1:** Create variables of different types and assign appropriate values.

*[Copy the snapshot (showing memory, variables and objects) and attach here]*

* Discuss about the amount of memory allocated to each type: byte, int, short, boolean, etc.

*[Write your answer here]*

**Exercise-2:** Create a variable of pointer type.

* Discuss the type of value a pointer can store.

*[Write your answer here]*

* Make it to refer to (point to) any of the variables you have created.

*[Copy the snapshot (showing memory, variables and objects) and attach here]*

**Exercise-3:** Create an integer array (size 5, values 10, 20, 30, 40 and 50).

*[Copy the snapshot (showing memory, variables and objects) and attach here]*

* Discuss how array elements are arranged in memory.

*[Write your answer here]*

* What is the amount of memory allocated to the array?

*[Write your answer here]*

**Exercise-4:** The following classes (Foo and Bar) are defined:

A screenshot of a computer code

Description automatically generated

* Draw a box-and-pointer diagram showing the program state after the following Python code has been executed:

A close-up of text

Description automatically generated

*[Write your answer here Draw/Include a BPD ]*

* Write the necessary python code that when executed results in the stated as reflected in the following BPD:

A diagram of a diagram

Description automatically generated

*[Write your answer here Draw/Include necessary python code]*

*plugh = Foo(8,9)*

*xyzzy = Bar(plugh,None)*

*thud = Bar(xyzzy,xyzzy)*

**Exercise-5:** Linked List Operations (Implementation)

As discussed, our linked list is a collection of nodes where node is:

A diagram of a data point

Description automatically generated

Each has a data and a pointer(reference) to the next node of the list. We will call this field as next.

#Define Node Class

class Node:

def \_\_init\_\_(self, data):

self.data = data # Data

self.next = None # Pointer

1. We define the class for Node.

The first of a linked list is known as the **head** the list. By starting at the **head**, and moving from one node to another by following each node’s next reference, we can traverse the whole list.

1. We define the class for LinkedList.

#Define Linked List Class

class LinkedList:

def \_\_init\_\_(self):

# When list is created, it empty, head will point to null

self.head = None

1. Add, Delete Operations

We can add and delete nodes anywhere within the list. For now, we will add and delete at the front.

def add\_front(self, data):

new\_node = Node(data)

new\_node.next = self.head

self.head = new\_node

def delete\_front(self):

if self.head:

self.head = self.head.next

1. We will write a display method where we traverse the list and display the data of each node.

def display(self):

current = self.head

while current:

print(current.data, end=" -> ",)

current = current.next

1. Let us create our list and perform various operations.

linked\_list = LinkedList()

linked\_list.add\_front(1)

linked\_list.add\_front(2)

linked\_list.add\_front(3)

linked\_list.display()

linked\_list.delete\_front()

linked\_list.display()

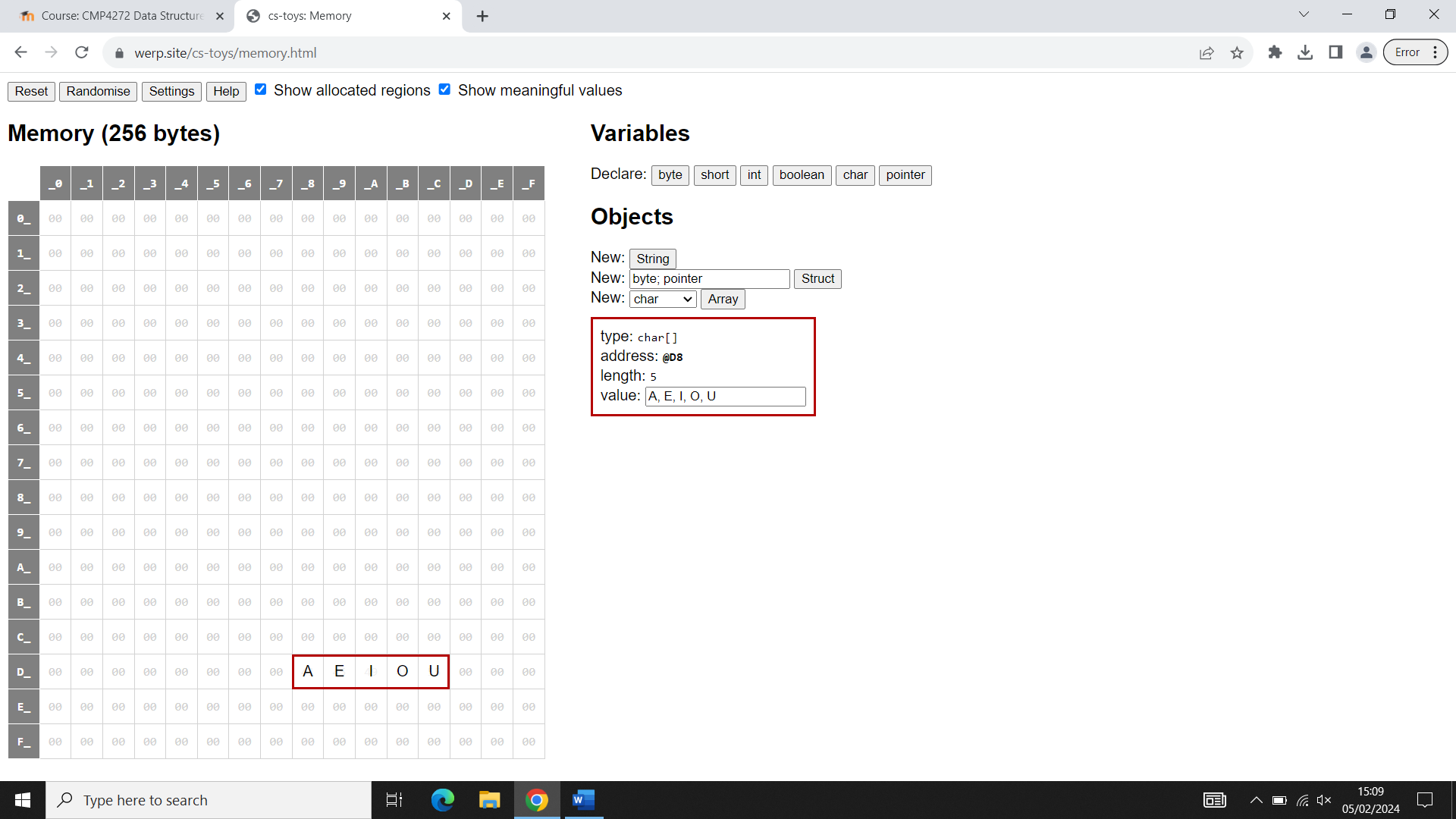
**TODO** : To add/delete a node at the end or anywhere in the list.

1. **Lab Submission Exercises:**

* **Submitting the solution of the following exercises is mandatory.**
* **Solutions that comprise of python code, must be well documented. (Include necessary comments)**

**For Exercise-6 and 7, use the** [**memory visualization tool**](https://werp.site/cs-toys/memory.html)**.**

**Exercise-6:** Create a char array (size 5, values A, E, I, O and U).



* Discuss how array elements are arranged in memory.

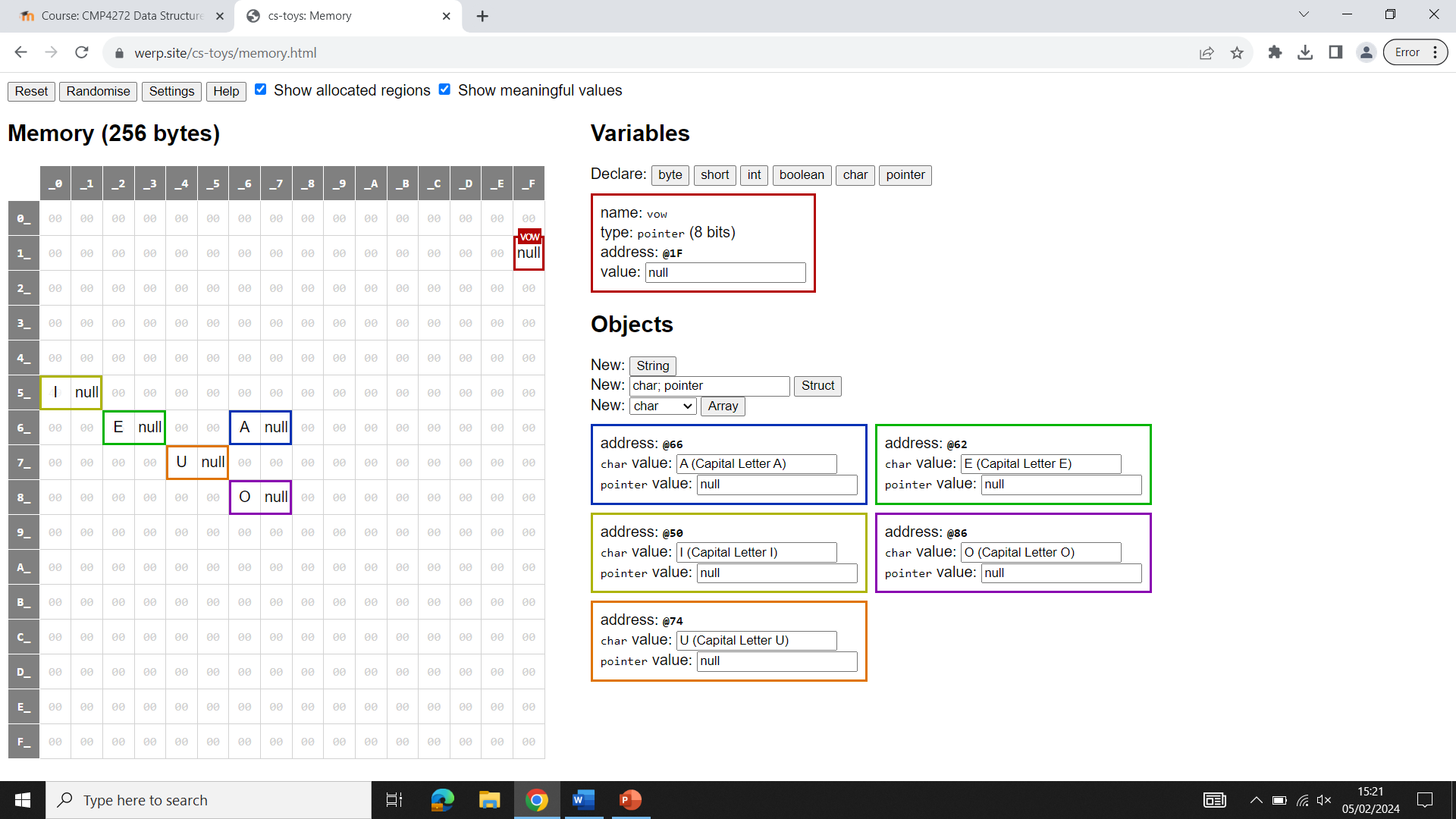
Arrays are expandable. The limit of an array expands when a new element is being added to a full array. The maximum length of an array doubles each time it expands.

* What is the amount of memory allocated to the array?

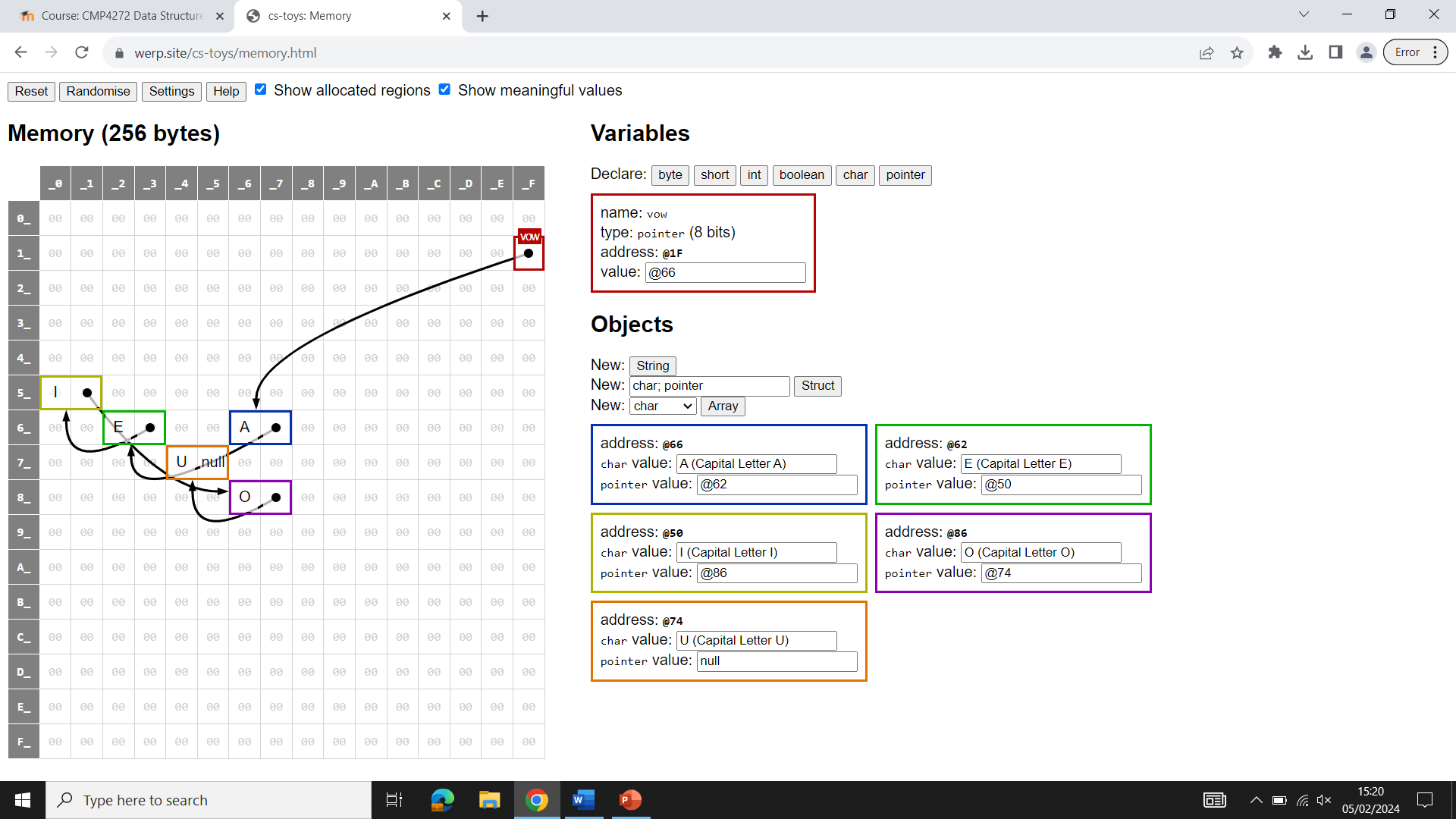
40 Bits – 5 Bytes

**Exercise-7:** We want to store vowels in a linked list, address of which is stored in a pointer vow.

* Create appropriate variables and objects.



* Create the proper linkages to form a linked list.



* Discuss what each of the pointer(reference) fields hold.

The pointer holds the reference address of another part in the memory. For example, in the image above, the pointer “vow” has the value @66. This links the pointer to the character “A” located at 66.

**Exercise-8:** The following classes (Foo and Bar) are defined:

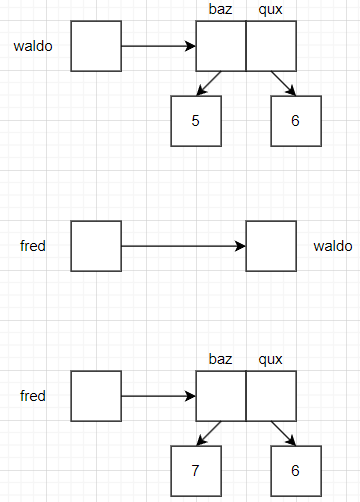
A screenshot of a computer code

Description automatically generated

* Draw a box-and-pointer diagram showing the program state after the following Python code has been executed:

A white rectangular box with black text

Description automatically generated

****

* Write the necessary python code that when executed results in the state as reflected in the following BPD:

A diagram of a flowchart

Description automatically generated

*[Write your answer here Draw/Include necessary python code]*

**Exercise-9:** Refer to the Linked List implementation (Exercise 5), write the functions to add and delete a node at the end of the linked list. Test your code.

*[Write your answer here Draw/Include necessary python code and output]*

1. **Moodle Submission:**

You are required to submit your solution in the word document.

Naming Format: **StudetName\_studentID.docx [ or other word formats]**

Example : AliceSmith\_514099.docx

**NOTE**

* It is important to complete the weekly labs in particular labs 2, 3, 4, 5 and 6 because it contains questions that are part of the coursework. (Weightage: 25%).
* Only one of these labs will be chosen randomly for marking, so it is important that you complete and submit each of these labs.
* You must submit each lab within one week of that lab session.
* If you are unable to finish all the tasks, submit whatever you’ve managed to produce by the due date.
* Solutions that comprise of python code, must be well documented. (Include necessary comments)