Supplemental Materials
Arrays and Linked Lists

Java Arrays

1. Must be declared and memory allocated of some integer size:

```
int [] numbers = new int[5];  // array of size 5
```

2. Can have primitive or object elements:

```
String [] names;
```

3. Stored as object in Java so has length member:

```
numbers.length;  // returns size of array in this case 5
```

4. First element is at index 0

```
numbers[0] = 10;
```

5. To traverse typically use for-loop or foreach loop

```
for (int i=0; i < numbers.length; i++)
        numbers[i] = 0;
```

6. Accessing illegal index causes **ArrayIndexOutOfBoundsException**

Java Linked List

1. Create node with data type you want to store and next node:

```
class Node  {
    private Integer value;   // data
    private Node next;       // next node

    // implement get and set methods
}
```

2. Linked list needs head, tail, and methods to insert and delete and any others as needed. Also need to initialize list in constructor

```
class SingleLinkedList {
    private Node head;
```

```
    private Node tail;

    public SingleLinkedList() {
            head = tail = null;
    }

    public void insert(Integer data) {
                // implementation
    }


    public void delete(Integer data) {
                // implementation
    }

    // set and get methods

}
```

C++ Arrays

1. Must be declared for some integer size:

```
int numbers[5];  // array of size 5
```

2. Can have primitive or object elements:

```
Employee [] names;   // assumes there is class defined for Employee
```

3. Length of the array has to be stored as variable so need to pass the value to methods

4. First element is at index 0

```
numbers[0] = 10;
```

5. To traverse typically use for-loop or foreach loop

```
for (int i=0; i < numbers.length; i++)
        numbers[i] = 0;
```

6. Accessing illegal index causes **undefined behavior** depending what is stored at that particular memory location

C++ Linked List

1. Create node of the type you want to store:

```
struct node  {
      int value;   // data
      struct node *next;       // pointer to next node
}
```

2. Linked list needs head, tail, and methods to insert and delete and any others as needed

```
class singleLinkedList {
      private:
            node *head, *tail;

      public:

            singleLinkedList() {
                  head = tail = NULL;
            }

            void insert(int data) {
                 // implementation
            }


            void delete(int data) {
                 // implementation
            }

}
```

Python Arrays

1. Array in Python is a list and is not fixed length nor does it have to be declared. To create an empty list:

```
numbers = []  # array which is list in Python of size 0
```

2. Python does not require declaring data type and the data type is based on what is stored:

```
numbers. append(5)  # adds element 5 at end
numbers.insert(0, 4)  # inserts 4 at index 0
numbers.remove(4)  # searches for element with value 4 and removes it
numbers.pop(0) # removes element at index 0 and returns it
```

3. To find size of list use len:

len(numbers) # returns size of list

4. First element is at index 0

5. To traverse typically use for-in

```
for num in numbers
        num = 0
```

6. Accessing illegal index causes index error: **IndexError: list index out of range**

Python Linked List

1. Create class node:

```
class Node:
    def __init__(self,initdata):
        self.data = initdata
        self.next = None

    # set and get defs
}
```

2. Linked list needs head, tail, and methods to insert and delete and any others as needed

```
class singleLinkedList:
    def __init__(self):
        self.head = None
        self.tail = None


    def insert(self, data):
        # implementation

    def delete(self, data) {
        # implementation
```