Question 1: What is a data structure? (part 1) Describe a linked list with examples. (part 2)

Q1P1)

Within computer science, a data structure is any one of many architectures for organizing data. Since computer science and software engineering have a wide range of problems that require a wide range of solutions for storing, organizing, and manipulating data, there are a large number of varying structures that fill within the scope of ‘data structures’. These all come in various forms, each with its own strengths and weaknesses. Common examples include arrays, linked lists, stacks, queues, trees, and hash tables. The choice of a data structure depends on the specific requirements of a problem or task. For instance, an array is ideal for quick access to elements by index, while a linked list excels in dynamic memory allocation.

Q1P2)

A linked list is a fundamental data structure in computer science consisting of a sequence of elements, called nodes, where each node stores a data element and a reference (or pointer) to the next node in the sequence. Unlike arrays, linked lists do not have a fixed size and can dynamically grow or shrink during runtime. This flexibility makes them suitable for situations where the size of the data structure needs to change frequently. Linked lists are commonly used in various applications, including implementing dynamic data structures like stacks and queues, and are valuable when memory allocation or deallocation needs to be efficient.

An example of a linked would be the following (2 -> 6 -> 7 -> 9) in which each node is a number, and each nodes pointer is pointing where the arrow to the right of each node is pointing. There will be further examples of linked lists and operations within them in the code output attached below for question 2.

Question 2: Write a program to implement different operations on a linked list. (Program in linked-list.py, output attached below.)

```
nayatrov@ariahas-MBP hw3 % python3 linked-list.py
Enter a value (or any non-integer to stop): 4
Enter a value (or any non-integer to stop): 6
Enter a value (or any non-integer to stop): 333
Enter a value (or any non-integer to stop): 2
Enter a value (or any non-integer to stop): 5
Enter a value (or any non-integer to stop):
Now printing Linked List: 'End' will indicate the prior element was the last within the list.
4 -> 6 -> 333 -> 2 -> 5 -> End
Linked List after removing the first element:
6 -> 333 -> 2 -> 5 -> End
Enter the position to insert (0 for head): 2
Enter the value to insert: 444
Linked List after insertion:
6 -> 333 -> 444 -> 2 -> 5 -> End
Now creating linked list using the sum of two other linked lists to fill the data:
Enter values for the first linked list (enter a non-integer to stop):
4
7
3
11

Enter values for the second linked list (enter a non-integer to stop):
34
654
77
66

First Linked List:
4 -> 7 -> 3 -> 11 -> End
Second Linked List:
34 -> 654 -> 77 -> 66 -> End
Third Linked List (Sum of the First and Second Lists):
38 -> 661 -> 80 -> 77 -> End
Fourth Linked List (Union of First and Second Lists):
4 -> 7 -> 3 -> 11 -> 34 -> 654 -> 77 -> 66 -> End
Fifth Linked List (Intersection of First and Second Lists):
End
```