In-Class Exercise - Part 1: https://github.com/hien01604/dsa/blob/main/22520422_classwork_1

1. Find the Middle Node of a Linked List

```
== Input List 1 ==
Number of nodes: 5
Input List 1: 1 2 3 4 5
List 1: 1 2 3 4 5

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8: 1
The middle node: 3
```

2. Detect a Cycle in a Linked List

```
== Input List 1 ==
Number of nodes: 4
Input List 1: 1 2 3 4
List 1: 1 2 3 4

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8: 2
List doesn't have a cycle.
```

3. Combine Two Sorted Linked Lists

```
== Input List 1 ==
Number of nodes: 4
Input List 1: 1 2 3 4
List 1: 1 2 3 4

== Input List 2 ==
Number of nodes: 3
Input List 2: 5 6 7
List 2: 5 6 7

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8: 3
Merged List: 1 2 3 4 5 6 7
```

4. Reverse a Linked List

```
== Input List 1 ==
Number of nodes: 4
Input List 1: 1 2 3 4
List 1: 1 2 3 4

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8: 5
Reversed List: 4 3 2 1
```

5. Eliminate Duplicates from a Sorted Linked List

```
== Input List 1 ==
Number of nodes: 4
Input List 1: 1 2 3 3
List 1: 1 2 3 3

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8:36
List after removing duplicates: 1 2 3
```

6. Check if a Linked List is a Palindrome

```
== Input List 1 ==
Number of nodes: 5
Input List 1: 1 2 3 2 1
List 1: 1 2 3 2 1

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8: 7
The list is Palindrome.
```

```
== Input List 1 ==
Number of nodes: 4
Input List 1: 1 2 3 4
List 1: 1 2 3 4

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8: 7
The list is not Palindrome.
```

8. Write a function to search for nodes with the value X in the list. If found, return the addresses of the nodes; if not found, return NULL.

```
== Input List 1 ==
Number of nodes: 4
Input List 1: 1 2 3 4
List 1: 1 2 3 4

===== MENU =====
0. Exit.
1. Find the middle node of the list.
2. Detect a cycle in a linked list.
3. Combine Two Sorted Linked Lists.
4. Find the Intersection of Two Linked Lists.
5. Reverse a Linked List.
6. Eliminate Duplicates from a Sorted Linked List.
7. Check if a Linked List is a Palindrome.
8. Search for nodes with the value X in the list.
Choose a number 0-8: 8
Enter value to search: 2
The node value 2: 0x14286f0
```

In-Class Exercise - Part 2: https://github.com/hien01604/dsa/blob/main/22520422_classwork_2

1. Add Two Numbers

```
Enter numbers (end with 0): 1 2 3 4 0

Choose option:
1. Add two numbers to the list
2. Copy list with random pointers
3. Swap two nodes
4. Remove from end
5. Separate odd and even
6. Divide into parts
7. Remove zero-sum consecutive nodes
8. Automatic input
0. Exit
Enter your choice: 1
Enter first number: 5
Enter second number: 6
1 2 3 4 5 6
```

2. Copy List with Random Pointers
3. Swap Nodes in a Linked List

```
1 2 3 4 5 6

Choose option:
1. Add two numbers to the list
2. Copy list with random pointers
3. Swap two nodes
4. Remove from end
5. Separate odd and even
6. Divide into parts
7. Remove zero-sum consecutive nodes
8. Automatic input
0. Exit
Enter your choice: 3
Enter two values to swap: 2 6
List after swapping nodes with 2 and 6:
1 6 3 4 5 2
```

4. Remove the N-th Node from the End of a List

```
1 6 3 4 5 2

Choose option:
1. Add two numbers to the list
2. Copy list with random pointers
3. Swap two nodes
4. Remove from end
5. Separate odd and even
6. Divide into parts
7. Remove zero-sum consecutive nodes
8. Automatic input
0. Exit
Enter your choice: 4
Enter the position from the end to remove: 3
List after removing (from the end):
1 6 4 5 2
```

5. Separate Odd and Even Nodes in a Linked List

```
1 6 4 5 2

Choose option:
1. Add two numbers to the list
2. Copy list with random pointers
3. Swap two nodes
4. Remove from end
5. Separate odd and even
6. Divide into parts
7. Remove zero-sum consecutive nodes
8. Automatic input
0. Exit
Enter your choice: 5
List after separating odd and even:
1 5 6 4 2
```

6. Divide a Linked List into Parts

```
1 5 6 4 2

Choose option:
1. Add two numbers to the list
2. Copy list with random pointers
3. Swap two nodes
4. Remove from end
5. Separate odd and even
6. Divide into parts
7. Remove zero-sum consecutive nodes
8. Automatic input
0. Exit
Enter your choice: 6
Enter number of parts: 2
Part 1: 1 5 6
Part 2: 4 2
```

7. Remove Zero-Sum Consecutive Nodes from a Linked List

```
Enter numbers (end with 0): 1 -2 2 3 4 -4 0

Choose option:
1. Add two numbers to the list
2. Copy list with random pointers
3. Swap two nodes
4. Remove from end
5. Separate odd and even
6. Divide into parts
7. Remove zero-sum consecutive nodes
8. Automatic input
0. Exit
Enter your choice: 7
List after removing zero-sum consecutive nodes:
1 3
```

8. Write a function to input values for a list using the automatic input method, with values selected from the range [-99; 99]. The number of entries is randomly chosen from the range [39; 59] (using a function to insert at the end of the list)

```
1 3

Choose option:
1. Add two numbers to the list
2. Copy list with random pointers
3. Swap two nodes
4. Remove from end
5. Separate odd and even
6. Divide into parts
7. Remove zero-sum consecutive nodes
8. Automatic input
0. Exit
Enter your choice: 8
List after automatic input:
1 3 93 48 -47 -6 15 45 54 -31 -90 36 57 -31 41 -45 -66 -18 -75 -45 27 72 22 90 -41
    -76 -33 38 -66 84 -15 -97 58 -45 51 -89 48 -34 32 -20 11 19 92 -54 87
```