Implementation of a doubly linked list in Kotlin. Here is a breakdown of the code and its functionality:

1. Node Class: It represents an individual node in the doubly linked list. Each node contains a data value and two pointers: `next` pointing to the next node and `prev` pointing to the previous node.

2. DoublyLinkedList Class: It represents the doubly linked list itself and contains various operations to manipulate the list.

- Append Operation: The `append` function adds a new node at the end of the list. It creates a new node, updates the `next` and `prev` pointers accordingly, and handles the edge case when the list is empty.

- Push Operation: The `push` function adds a new node at the beginning (head) of the list. It creates a new node, updates the `next` and `prev` pointers accordingly, and handles the case when the list is not empty.

- Insert After Operation: The `insertAfter` function adds a new node in the middle of the list, after a given node. It creates a new node, updates the `next` and `prev` pointers of the adjacent nodes and the new node itself.

- Delete Operation: The `delete` function removes a specified node from the list. It updates the `next` and `prev` pointers of the adjacent nodes to bypass the deleted node.

- Length Operation: The `length` function returns the total number of nodes in the list by traversing through each node and counting them.

- Search Operation: The `search` function searches for a node with a given data value. It traverses through each node and compares the data value with the search value.

- GetNode Operation: The `getNode` function retrieves a node at a specific index in the list. It traverses through the list until it finds the desired index and returns the node.

3. Main Function: It demonstrates the usage of the doubly linked list by creating an instance of `DoublyLinkedList`, performing various operations, and printing the results.

The code appends 6 to the list, pushes 7 and 1 to the front, appends 4, inserts 8 after the second node, prints the length of the list, and searches for elements 8 and 10. Then, it deletes the first and third nodes, and finally prints the updated length of the list.

The code showcases common operations of a doubly linked list like appending, pushing, inserting after, deleting, calculating length, and searching for a node.