Project Description:

The project is about implementing a **Doubly Linked List** (DLL), which is a linked data structure where each node contains a reference to the next node and the previous node. This allows traversal in both directions.

Attributes:

1. **Node Class:**
    - `value` : Stores the data of the node.
    - `next` : Pointer to the next node.
    - `prev` : Pointer to the previous node.
2. **DoublyLinkedList Class:**
    - `head` : Pointer to the first node.
    - `tail` : Pointer to the last node.
    - `size` : The number of nodes in the list.

Methods:

1. **add_to_front(value)**: Adds a new node at the beginning.
2. **add_to_end(value)**: Adds a new node at the end.
3. **remove_from_front()**: Removes the node from the front.
4. **remove_from_end()**: Removes the node from the end.
5. **find(value)**: Searches for a node by value.
6. **insert_at(index, value)**: Inserts a node at a given index.
7. **remove_at(index)**: Removes a node at a given index.
8. **get_size()**: Returns the size of the list.
9. **reverse_traversal()**: Prints the list in reverse order.
10. **print_list()**: Prints the list from front to back.
11. **check_empty()**: Checks if the list is empty.
12. **clear_list()**: Clears all the nodes in the list.
13. **get_at(index)**: Retrieves the value at a given index.
14. **swap_nodes(index1, index2)**: Swaps two nodes at specified indices.
15. **detect_cycle()**: Detects if the list has a cycle.

Main Method:

The main method will instantiate the Doubly Linked List, perform various operations like adding/removing elements, and validate them using if-else conditions to ensure the correct pointers (`next` and `prev`) are maintained.