1. Name of the Project
Double Linked List Manipulations

2. List Authors of the Project
Alex Straight

3. File Structure of the project submitted
Alex_Straight.zip -> DoubleLinkedList.cpp ReadMe.pdf

4. Software that can be used to run the files / Installation Guide
Software: A C++ compiler (like g++ or clang++).
Steps:
Install a C++ compiler (if not already installed).
Navigate to the directory where double_linked_list.cpp is located.
Compile the code using g++ double_linked_list.cpp -o output_name.
Run the program using ./output_name.
5. How to get started?
a. Describe the input required from the user
The user will be provided with a menu of choices for various linked list operations. Depending on the option selected, they may be prompted to enter more data (like index, value, and name).

b. Describe what function is called by each input and briefly describe each function.

1. Create new list: Initializes a new list with a starting element.
2. Append data: Appends data to the end of the list.
3. Prepend data: Adds data to the beginning of the list.
4. Insert data at index: Inserts data at a specific index.
5. Print list: Displays the entire list.
6. Delete from head: Removes the first element of the list.
7. Delete from tail: Removes the last element of the list.
8. Delete at index: Deletes data at a specific index.
9. Sort list: Sorts the list in ascending order based on the value.
10. Reverse list: Reverses the order of elements in the list.
11. Count multiples of a value: Counts the number of nodes with a specified value.
12. Remove multiples: Removes nodes with duplicate values.
13. Split list: Splits the list into two separate lists.
0. Exit: Exits the program.

6. Runtime: Derive the Big O of each function mathematically or by recognizing the section of code that takes the maximum time to execute.

append: O(1) - Direct addition to the tail.
prepend: O(1) - Direct addition to the head.
insert: O(n) - Worst case when inserting at the end.
printList: O(n) - Traversing and printing the entire list.
deleteAtHead: O(1) - Direct deletion of head.
deleteAtTail: O(1) - Direct deletion of tail.
deleteAtIndex: O(n) - Worst case when deleting from the end.
sortList: O(n^2) - Bubble sort is applied for sorting.
reverseList: O(n) - List is traversed once.
countMultiples: O(n) - List is traversed once for counting.
removeMultiples: O(n^2) - Uses countMultiples for each node.

headTailSplit: O(n) - Traversal using slow and fast pointers.