Student name: Vikram Thangavel
Student ID: 216933327
EECS login ID: vikram19

**Assignment 6 Report**

**Instruction**: Complete the report then convert it to PDF to submit.

**Academic Honesty Pledge**: I affirm that I have not given or received any unauthorized help in this assignment, and that this work is my own. Any authorized references are acknowledged below.

**Sign or type your name here**: __Vikram Thangavel_____

For each function to be implemented, fill in the information using the following template.

**Function Name: insertFirst()**

- Reference(s): https://www.geeksforgeeks.org/doubly-linked-list/
- https://www.tutorialspoint.com/data_structures_algorithms/doubly_linked_list_program_in_c.htm
- https://www.youtube.com/watch?v=jkbS-bKXTEE
- Error conditions: If node can't be created, an error message is printed "Insufficient memory" and returns null.
- Description of the code (algorithm): A node is created and checks if it's created or not. If it is, the new node's data value is assigned. Then, the new node's next is made as the head node's next. Then, the new node's previous is made as head. Then, it checks if the new node's next points to null. If it doesn't, then, the new node's next previous is changed. Afterwards, it returns the pointer of the new node that is added.
- Running time of the function (algorithm): O(1)
- Explanation of the running time: As there is no loop, the running time of the function is O(1) since the pointers are only changed around.

**Function Name: insertLast()**

- Reference(s): https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/
- https://www.javatpoint.com/insertion-in-doubly-linked-list-at-the-end#:~:text=In%20order%20to%20insert%20a,the%20new%20node%20being%20inserted
- Error conditions: If node can't be created, an error message is printed "Insufficient memory" and returns null.
- Description of the code (algorithm): A node is created and checks if it's created or not. If it is, the new node's data value is assigned. Then, the new node's previous is made as tail's previous. Then, the new node's next points to tail. Then, it checks if insert's previous points to null. If it doesn't, then the new node previous next points to the new node. Otherwise, head is changed to new node. Afterwards, the pointer of the new node is returned.
- Running time of the function (algorithm): O(1)
- Explanation of the running time: As there is no loop, the running time of the function is O(1) since the pointers are only changed around.

**Function Name: removeFirst()**

- Reference(s): https://www.geeksforgeeks.org/delete-a-node-in-a-doubly-linked-list/
- https://www.javatpoint.com/deletion-in-doubly-linked-list-at-beginning
- Past assignments (Assignment 4)
- Error conditions: If list is empty which means only dummy nodes, it will print an error message "Empty list!" and returns -1.
- Description of the code (algorithm): A new node is created which is the node that is going to be removed. If the list is empty (nothing except dummy nodes), error message is printed and -1 is returned. If the list is not empty, then the new node is made as head. Then, head is made as head pointing to next. Afterwards, the node's data is assignment to a return int variable and is removed (freed). Then, the data of node is returned.
- Running time of the function (algorithm): O(1)
- Explanation of the running time: As there is no loop, the running time of the function is O(1) since the pointers are only changed around.

**Function Name: removeLast()**

- Reference(s): https://www.geeksforgeeks.org/linked-list-set-3-deleting-node/
- https://www.geeksforgeeks.org/delete-a-node-in-a-doubly-linked-list/ (Diagram)
- Error conditions: If list is empty which means only dummy nodes, it will print an error message "Empty list!" and returns -1.
- Description of the code (algorithm): The data of the node that is going to be removed is assigned to an int variable. Then afterwards, we want to break the pointers to the second last node and assign them to the third last and last. So, the tail's previous previous next pointer is pointed to the tail node. Then, the tail's previous is made as tail's previous previous. As the second last node has no more pointers, it is removed.
- Running time of the function (algorithm): O(1)
- Explanation of the running time: As there is no loop, the running time of the function is O(1) since the pointers are only changed around.

**Function Name: search()**

- Reference(s): Previous assignments (Assignment 4)
- Error conditions: If the list does not contain the non-negative integer k, the function will return null.
- Description of the code (algorithm): A node is created and inserted in the chain of nodes. Then, the element takes the first node's location and searches for a node that contains the element k and returns its address.
- Running time of the function (algorithm): O(n)
- Explanation of the running time: Function performs a linear search to look for a node that contains the element k so therefore, the function's running time is O(n).