Assignment 2 - Muzan Daffaallah

Part 1:

Running the code:

START. Create a new instance of Single_Linked_List for integers. Display a menu with options. Get the user's choice from the menu. If the user chooses to add to the front, ask for a number, call push_front with the number, print message. If the user chooses to add to the back, ask for a number, call push_back with the number and print message. If the user chooses to remove from the front, call pop_front, print message. If the user chooses to remove from the back call pop_back and print message. If the user chooses to insert at position, ask for a number and a position, call insert with a number and position, print message. If the user wants to find an item, ask for a number, call find with number, if found print found message, if not print not found message. If the user chooses to print the list, call printList and print all items in the list. If the user chooses to exit, print message and break loop. **END.**

Screenshots:

Additions: I've added a menu for efficiency and a print option to display the progress of the list as we alter it.

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice:
```

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 8
List: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> nullptr
```

1. push_front

Element will be pushed to the front of the list

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 1
Enter a value to push to the front: 12
Value added to the front.
```

```
List: 12 -> nullptr
```

2. push_back

Element will be pushed to the back

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 2
Enter a value to push to the back: 10
Value added to the back.
```

```
List: 12 -> 10 -> nullptr
```

3. pop_front

Element is removed from the front of the list

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 3
Front element removed.
```

```
List: 10 -> nullptr
```

4. pop_back

Element was removed from the back of the list

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 4
Back element removed.
```

```
List is empty.
```

5. Front

a. If the list is not empty then the integer at the front of the list will be returned

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 9
Front element: 10
```

b. If the list is empty a message will be printed

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 9
List is empty.
```

6. Back

a. If the list is not empty then the integer at the front of the list will be returned

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 10
Back element: 10
```

b. If the list is empty a message will be printed

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 10
List is empty.
```

7. Empty

a. If the list is empty it will return “empty” message

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 11
Empty
```

b. If the list is not empty it will return is “not empty” message

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 11
Not Empty
```

8. Insert Index

Given an index, an element can be inserted at a specific location in the list

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 5
Enter the index to insert at: 6
Enter the value to insert: 100
Element inserted at index 6.
```

```
List: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 100 -> 7 -> 8 -> 9 -> 10 -> nullptr
```

9. Remove with index

Given the index, the program will remove the element at that index.

a. If the index is in range the element will be removed

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 6
Enter the index to remove from: 4
Element removed from index 4.
```

b. If the index is out of range an error message will be returned

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 6
Enter the index to remove from: 11
Invalid index.
```

10. Find element given index

Given the index, find the first occurrence of the element.

a. If the element is found “found” message is returned with index

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 7
Enter the value to find: 6
Value found at index 5.
```

b. If the element is not found “not found” message is returned

```
Menu:
1.  Push Front
2.  Push Back
3.  Pop Front
4.  Pop Back
5.  Insert at Index
6.  Remove from Index
7.  Find an Element
8.  Print List
9.  Show Front Element
10. Show Back Element
11. Check Empty Status
12. Exit
Enter your choice: 7
Enter the value to find: 11
Value not found.
```

Part 2:

Running the code:

START. Create a pointer and set it to null. Display the menu. If the user chooses 1, create the stack. If they choose 2, ask for a number and push it onto the stack. If they choose 3, pop the top element. If they choose 4, display the top element of the stack. If they choose 5, display the average of the element in the stack, and if there are no items in the stack return an error message. If they choose 6, check if the stack is empty and return true or false. If they choose 7, display all of the elements in the stack. If they choose 8, exit the program. **END.**

Additions: I’ve added a menu and a print the display for efficiency.

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice:
```

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 7
List: 56 -> 34 -> 23 -> 12 -> nullptr
```

1. Before you can continue with any other operations, the program verifies that the stack was created.

a. If the stack was created a verification message is printed

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 1
Stack created successfully!
```

b. If not an error message will be printed

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 2
Please create a stack first (Option 1).
```

2. Check to see if the stack is empty or not.

a. If the stack is empty, “empty” message is printed.

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 6
Stack is empty.
```

b. If the stack is not empty, “not empty” message is printed.

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 6
Stack is not empty.
```

3. To insert some integer values onto the stack, the integer is pushed onto the stack

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 2
Enter the value to push: 12
Pushed 12 onto the stack.
```

4. To remove an element, the element is popped from the stack

a. If there is an element in the stack it is popped

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 3
Popped: 12
```

b. If there is no element in the stack an error message is printed

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 3
Stack is empty. Cannot pop.
```

5. Print the top of the stack.

a. If there is an element on top of the stack it will be printed

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 4
Top element: 21
```

b. If no element in the stack, a message will be printed

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 4
Stack is empty. No top element.
```

6. Finding the average value of the stack

a. If the stack has values, the average will be returned.

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 5
Average of stack elements: 129.667
```

b. If no values in the stack, error message will be returned.

```
--- Stack Operations Menu ---
1. Create a stack
2. Push an element
3. Pop an element
4. Display top element
5. Display average of elements
6. Check if stack is empty
7. Display all elements in the stack
8. Exit
Enter your choice: 5
Average of stack elements: Stack is empty. No elements to average.
```