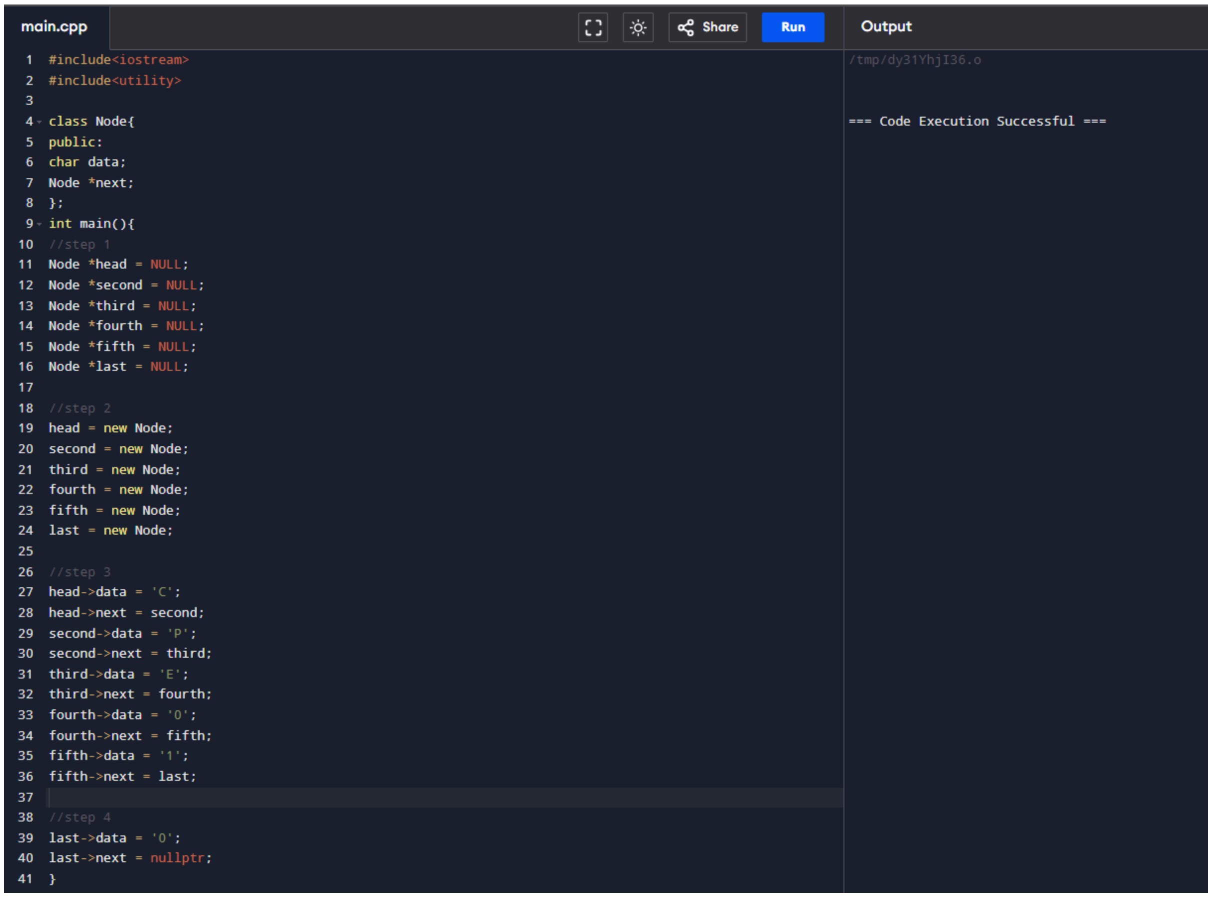
Activity No. 3	
LINKED LISTS	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 09/27/24
Section: CPE21S4	**Date Submitted:** 09/29/24
Name(s): Magistrado, Aira Pauleen M.	**Instructor:** Maria Rizette Sayo
6. Output	

Screenshot	

```cpp
#include<iostream>
#include<utility>

class Node{
public:
char data;
Node *next;
};
int main(){
//step 1
Node *head = NULL;
Node *second = NULL;
Node *third = NULL;
Node *fourth = NULL;
Node *fifth = NULL;
Node *last = NULL;

//step 2
head = new Node;
second = new Node;
third = new Node;
fourth = new Node;
fifth = new Node;
last = new Node;

//step 3
head->data = 'C';
head->next = second;
second->data = 'P';
second->next = third;
third->data = 'E';
third->next = fourth;
fourth->data = '0';
fourth->next = fifth;
fifth->data = '1';
fifth->next = last;

//step 4
last->data = '0';
last->next = nullptr;
}
```

```
/tmp/dy31YhjI36.o


=== Code Execution Successful ===
```

Discussion	**In the code a linked list was created but it does not display any output when it is run as it just sets up list and initializes and links nodes without displaying any output.**

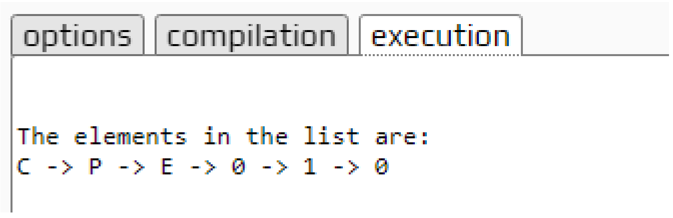
Table 3-1. Output of Initial/Simple Implementation

Operation	**Screenshot**
Traversal	

C/C++

```cpp
#include <iostream>
using namespace std;
```

```cpp
class Node {
public:
    char data;
    Node *next;
};

int main() {
   //step 1
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    //step 2
    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    //step 3
    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '0';
    fourth->next = fifth;

    fifth->data = '1';
    fifth->next = last;

    last->data = '0';
    last->next = NULL;

    Node *temp = head;
    printf("\n\nThe elements in the list are:  \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" -> ");
        }
    }

    return 0;
}
```

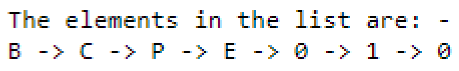


```
The elements in the list are:
C -> P -> E -> 0 -> 1 -> 0
```

Insertion at head

C/C++

```cpp
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
};

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '0';
    fourth->next = fifth;

    fifth->data = '1';
    fifth->next = last;

    last->data = '0';
    last->next = NULL;

    // Insertion at the head
    Node *newNode = new Node;
    newNode->data = 'B';
    newNode->next = head;
    head = newNode;

    Node *temp = head;
    printf("\n\nThe elements in the list are: - \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" -> ");
        }
    }

    return 0;
}
```



```
The elements in the list are: -
B -> C -> P -> E -> 0 -> 1 -> 0
```

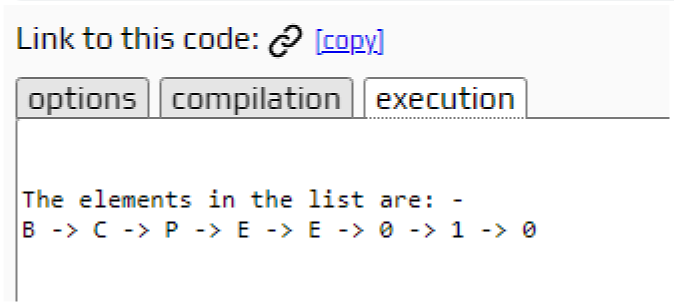
Insertion at any part of the list

C/C++

```
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
};

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;
```

```
head->data = 'C';
head->next = second;

second->data = 'P';
second->next = third;

third->data = 'E';
third->next = fourth;

fourth->data = '0';
fourth->next = fifth;

fifth->data = '1';
fifth->next = last;

last->data = '0';
last->next = NULL;

// Insertion at the head
Node *newNode = new Node;
newNode->data = 'B';
newNode->next = head;
head = newNode;

// Insertion at any position
int position = 4;
newNode = new Node;
newNode->data = 'E';

if (position == 1) {
    newNode->next = head;
    head = newNode;
} else {
    Node *temp = head;
    for (int i = 2; i < position; i++) {
        if (temp->next != NULL) {
            temp = temp->next;
        } else {
            cout << "Previous node cannot be null." << endl;
            delete newNode;
            return 1;
        }
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

// Traversal to print the list
Node *temp = head;
printf("\n\nThe elements in the list are: - \n");
while (temp != NULL) {
    printf("%c", temp->data);
    temp = temp->next;
    if (temp != NULL) {
        printf(" -> ");
```

```
            }
        }

        return 0;
    }
```

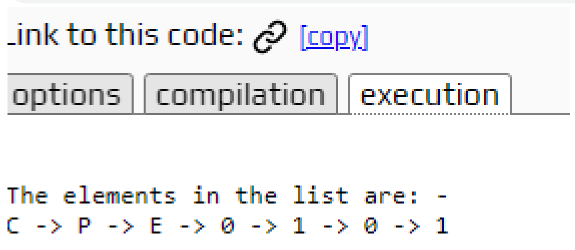

Link to this code: [copy]

options | compilation | execution

```
The elements in the list are: -
B -> C -> P -> E -> E -> 0 -> 1 -> 0
```

Insertion at the end

C/C++
```
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
};

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;
    fourth->data = '0';
    fourth->next = fifth;

    fifth->data = '1';
    fifth->next = last;

    last->data = '0';
    last->next = NULL;

    // Insertion at the end
    Node *newNode = new Node;
    newNode->data = '1';
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        Node *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;     }


    Node *temp = head;
    printf("\n\nThe elements in the list are: - \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" -> ");
        }
    }

    return 0;
}
```



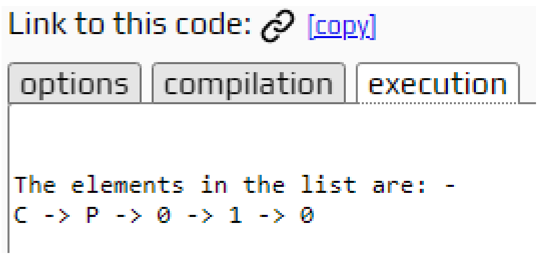
```
The elements in the list are: -
C -> P -> E -> 0 -> 1 -> 0 -> 1
```

Deletion of a node

```
C/C++
#include <iostream>
```

	```cpp
	using namespace std;

	class Node {
	public:
	    char data;
	    Node *next;
	};

	int main() {
	    Node *head = NULL;
	    Node *second = NULL;
	    Node *third = NULL;
	    Node *fourth = NULL;
	    Node *fifth = NULL;
	    Node *last = NULL;

	    head = new Node;
	    second = new Node;
	    third = new Node;
	    fourth = new Node;
	    fifth = new Node;
	    last = new Node;

	    head->data = 'C';
	    head->next = second;

	    second->data = 'P';
	    second->next = third;

	    third->data = 'E';
	    third->next = fourth;

	    fourth->data = '0';
	    fourth->next = fifth;

	    fifth->data = '1';
	    fifth->next = last;

	    last->data = '0';
	    last->next = NULL;

	    int position = 3;

	    Node *temp = head;

	    if (position == 1) {
	        head = head->next;
	        delete temp;
	    } else {
	        for (int i = 2; i < position; i++) {
	            temp = temp->next;
	        }
	        Node *nodeToDelete = temp->next;
	        if (nodeToDelete != NULL) {
	            temp->next = nodeToDelete->next;
	```

```
            delete nodeToDelete;
        }
    }


    Node *printTemp = head;
    printf("\n\nThe elements in the list are: - \n");
    while (printTemp != NULL) {
        printf("%c", printTemp->data);
        printTemp = printTemp->next;
        if (printTemp != NULL) {
            printf(" -> ");
        }
    }

    return 0;
}
```



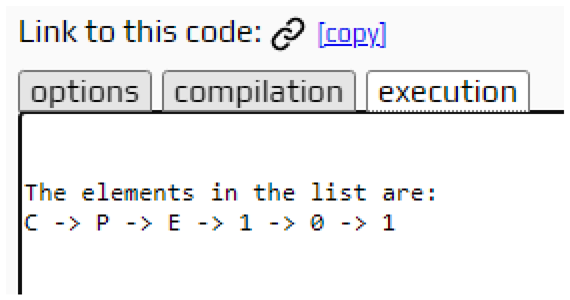
```
The elements in the list are: -
C -> P -> 0 -> 1 -> 0
```

Table 3-2. Code for the List Operations

a. **Source code**

```
C/C++
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
};

void traverseList(Node *head) {
    Node *temp = head;
    printf("\n\nThe elements in the list are:  \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" -> ");
        }
    }
}

int main() {
    // Step 1
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    // Step 2
    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    // Step 3
    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '1';
    fourth->next = fifth;

    fifth->data = '0';
    fifth->next = last;

    last->data = '1';
    last->next = NULL;

    traverseList(head);

    return 0;
}
```

Console



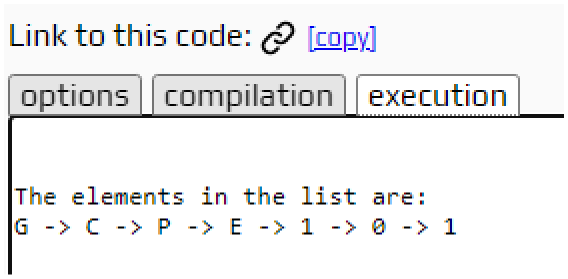
```
The elements in the list are:
C -> P -> E -> 1 -> 0 -> 1
```

b.	**Source code**	C/C++ (code below)

C/C++

```
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
};

void traverseList(Node *head) {
    Node *temp = head;
    printf("\n\nThe elements in the list are:  \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" -> ");
        }
    }
}

int main() {
    // Step 1
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    // Step 2
    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    // Step 3
    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '1';
    fourth->next = fifth;

    fifth->data = '0';
    fifth->next = last;
    last->data = '1';
    last->next = NULL;

    // Insertion at the head
    Node *newNode = new Node;
    newNode->data = 'G';
    newNode->next = head;
    head = newNode;

    traverseList(head);

    return 0;
}
```

Console



```
The elements in the list are:
G -> C -> P -> E -> 1 -> 0 -> 1
```

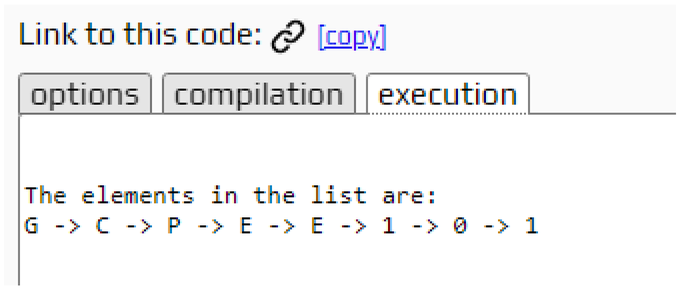
c. **Source code**

C/C++

```
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
};

void traverseList(Node *head) {
    Node *temp = head;
    printf("\n\nThe elements in the list are:  \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" -> ");
        }
    }
}

int main() {
    // Step 1
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    // Step 2
    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    // Step 3
    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '1';
    fourth->next = fifth;

    fifth->data = '0';
    fifth->next = last;

    last->data = '1';
    last->next = NULL;

    //  Insert at head
    Node *newNode = new Node;
    newNode->data = 'G';
    newNode->next = head;
    head = newNode;

    // Insert E
    Node *newNode2 = new Node;
    newNode2->data = 'E';

    Node *temp = head;
    while (temp != NULL) {
        if (temp->data == 'P') {

            newNode2->next = temp->next;
            temp->next = newNode2;
            break;
        }
        temp = temp->next;
    }

    traverseList(head);

    return 0;
```

```
}
```

Console



Link to this code: [copy]

options | compilation | execution

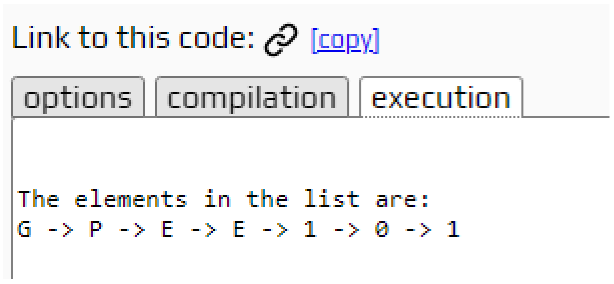
```
The elements in the list are:
G -> C -> P -> E -> E -> 1 -> 0 -> 1
```

d. **Source code**

C/C++

```cpp
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
};

void traverseList(Node *head) {
    Node *temp = head;
    printf("\n\nThe elements in the list are:  \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" -> ");
        }
    }
}

int main() {
    // Step 1
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    // Step 2
    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    // Step 3
    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '1';
    fourth->next = fifth;

    fifth->data = '0';
    fifth->next = last;

    last->data = '1';
    last->next = NULL;

    // Insert at head
    Node *newNode = new Node;
    newNode->data = 'G';
    newNode->next = head;
    head = newNode;

    // Insert E
    Node *newNode2 = new Node;
    newNode2->data = 'E';

    Node *temp = head;
    while (temp != NULL) {
        if (temp->data == 'P') {
            newNode2->next = temp->next;
            temp->next = newNode2;
            break;
        }
        temp = temp->next;
    }

    // Delete 'C'
    int position = 2;
    Node *temp2 = head;

    if (position == 1) {
        head = head->next;
        delete temp2;
    } else {
        for (int i = 2; i < position; i++) {
            temp2 = temp2->next;
        }
        Node *nodeToDelete = temp2->next;
        if (nodeToDelete != NULL) {
            temp2->next = nodeToDelete->next;
            delete nodeToDelete;
        }
    }

    traverseList(head);

    return 0;
}
```

Console



```
The elements in the list are:
G -> P -> E -> E -> 1 -> 0 -> 1
```

e. **Source code**

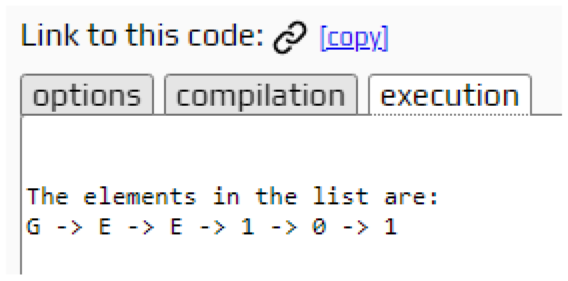
C/C++

```cpp
#include <iostream>

using namespace std;

class Node {
public:
    char data;
    Node* next;
};

void traverseList(Node* head) {
    Node* temp = head;
    cout << "\n\nThe elements in the list are: \n";
    while (temp != NULL) {
        cout << temp->data;
        temp = temp->next;
        if (temp != NULL) {
            cout << " -> ";
        }
    }
}

void insertAtStart(Node*& head, char data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = head;
    head = newNode;
}

void insertAfter(Node* prevNode, char data) {
    if (prevNode == NULL) {
```

		```
		        cout << "Previous node cannot be NULL." << endl;
		        return;
		    }

		    Node* newNode = new Node;
		    newNode->data = data;
		    newNode->next = prevNode->next;
		    prevNode->next = newNode;
		}

		void deleteNode(Node*& head, char data) {
		    if (head == NULL) {
		        cout << "List is empty." << endl;
		        return;
		    }

		    if (head->data == data) {
		        Node* temp = head;
		        head = head->next;
		        delete temp;
		        return;
		    }

		    Node* temp = head;
		    while (temp->next != NULL && temp->next->data != data) {
		        temp = temp->next;
		    }

		    if (temp->next == NULL) {
		        cout << "Node with data '" << data << "' not found." <<
		endl;
		        return;
		    }

		    Node* nodeToDelete = temp->next;
		    temp->next = nodeToDelete->next;
		    delete nodeToDelete;
		}

		int main() {
		    Node* head = NULL;
		    Node* second = NULL;
		    Node* third = NULL;
		    Node* fourth = NULL;
		    Node* fifth = NULL;
		    Node* last = NULL;

		    head = new Node;
		    second = new Node;
		    third = new Node;
		    fourth = new Node;
		    fifth = new Node;
		    last = new Node;

		    head->data = 'C';
		```

```
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '1';
    fourth->next = fifth;

    fifth->data = '0';
    fifth->next = last;

    last->data = '1';
    last->next = NULL;

    // Insert 'G' at the start
    insertAtStart(head, 'G');

    // Insert 'E' after 'P'
    Node* temp = head;
    while (temp != NULL && temp->data != 'P') {
        temp = temp->next;
    }
    if (temp != NULL) {
        insertAfter(temp, 'E');
    }

    // Delete 'C'
    deleteNode(head, 'C');

    // Delete 'P'
    deleteNode(head, 'P');

    // Traverse the list
    traverseList(head);

    return 0;
}
```

Console



```
The elements in the list are:
G -> E -> E -> 1 -> 0 -> 1
```

f.	**Source code**	

C/C++

```cpp
#include <iostream>

using namespace std;

class Node {
public:
    char data;
    Node* next;
};

void traverseList(Node* head) {
    Node* temp = head;
    cout << "\n\nThe elements in the list are: \n";
    while (temp != NULL) {
        cout << temp->data;
        temp = temp->next;
        if (temp != NULL) {
            cout << " -> ";
        }
    }
}

void insertAtStart(Node*& head, char data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = head;
    head = newNode;
}

void insertAfter(Node* prevNode, char data) {
    if (prevNode == NULL) {
        cout << "Previous node cannot be NULL." << endl;
        return;
    }

    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = prevNode->next;
    prevNode->next = newNode;
}

void deleteNode(Node*& head, char data) {
    if (head == NULL) {
        cout << "List is empty." << endl;
        return;
    }

    if (head->data == data) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }

    Node* temp = head;
    while (temp->next != NULL && temp->next->data != data) {
        temp = temp->next;
    }

    if (temp->next == NULL) {
        cout << "Node with data '" << data << "' not found." <<
endl;
        return;
    }

    Node* nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;
    delete nodeToDelete;
}

int main() {
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;
    Node* fourth = NULL;
    Node* fifth = NULL;
    Node* last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;

    second->data = 'P';
    second->next = third;

    third->data = 'E';
    third->next = fourth;

    fourth->data = '1';
    fourth->next = fifth;

    fifth->data = '0';
    fifth->next = last;

    last->data = '1';
    last->next = NULL;

    // Insert 'G' at the start
    insertAtStart(head, 'G');

    // Insert 'E' after 'P'
    Node* temp = head;
    while (temp != NULL && temp->data != 'P') {
        temp = temp->next;
```

		`    }`
		`    if (temp != NULL) {`
		`        insertAfter(temp, 'E');`
		`    }`
		
		`    // Delete 'C'`
		`    deleteNode(head, 'C');`
		
		`    // Delete 'P'`
		`    deleteNode(head, 'P');`
		
		`    // Traverse the list`
		`    traverseList(head);`
		
		`    return 0;`
		`}`

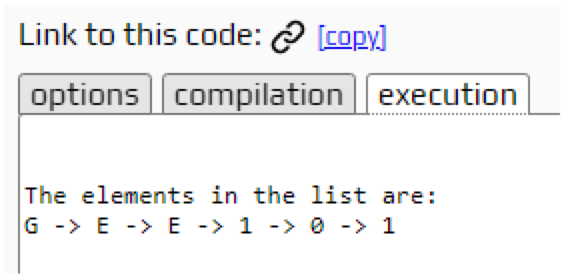
	Console	
		Link to this code: [copy]
		options compilation execution
		`The elements in the list are:`
		`G -> E -> E -> 1 -> 0 -> 1`

Table 3-3. Code and Analysis for Singly Linked Lists

Screenshots(s)	**Analysis**

C/C++
`#include <iostream>`
`using namespace std;`
 	
`class Node {`
`public:`
`    char data;`
`    Node *next;`
`    Node *prev;`
`};`
 	
`void traverseList(Node *head) {`
`    Node *temp = head;`
`    printf("\n\nThe elements in the list`
`are:  \n");`
`    while (temp != NULL) {`
`        printf("%c", temp->data);`
`        temp = temp->next;`
	- In the node there is a prev pointer added
	- The <-> displays the lists double linked list

```
        if (temp != NULL) {
            printf(" <-> ");
        }
    }
}

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;
    head->prev = NULL;

    second->data = 'P';
    second->next = third;
    second->prev = head;

    third->data = 'E';
    third->next = fourth;
    third->prev = second;

    fourth->data = '0';
    fourth->next = fifth;
    fourth->prev = third;

    fifth->data = '1';
    fifth->next = last;
    fifth->prev = fourth;

    last->data = '0';
    last->next = NULL;
    last->prev = fifth;

    traverseList(head);

    return 0;
}
```

- In the node there is a prev pointer added
- The <-> displays the lists double linked list

C/C++

```
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
    Node *prev;
};

void traverseList(Node *head) {
    Node *temp = head;
    printf("\n\nThe elements in the list
are: - \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" <-> ");
        }
    }
}

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;
    head->prev = NULL;

    second->data = 'P';
    second->next = third;
    second->prev = head;

    third->data = 'E';
    third->next = fourth;
    third->prev = second;

    fourth->data = '0';
    fourth->next = fifth;
    fourth->prev = third;

    fifth->data = '1';
    fifth->next = last;
    fifth->prev = fourth;

    last->data = '0';
    last->next = NULL;
    last->prev = fifth;

    Node *newNode = new Node;
    newNode->data = 'B';
    newNode->next = head;
    newNode->prev = NULL;
    head->prev = newNode;
    head = newNode;

    traverseList(head);

      return 0;
}
```

C/C++

```
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
    Node *prev;
};

void traverseList(Node *head) {
    Node *temp = head;
    printf("\n\nThe elements in the list
are: - \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" <-> ");
        }
    }
}

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
```

- In the node there is a prev pointer added
- The <-> displays the lists double linked list

```
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;
    head->prev = NULL;

    second->data = 'P';
    second->next = third;
    second->prev = head;

    third->data = 'E';
    third->next = fourth;
    third->prev = second;

    fourth->data = '0';
    fourth->next = fifth;
    fourth->prev = third;

    fifth->data = '1';
    fifth->next = last;
    fifth->prev = fourth;

    last->data = '0';
    last->next = NULL;
    last->prev = fifth;

    Node *newNode = new Node;
    newNode->data = 'B';
    newNode->next = head;
    newNode->prev = NULL;
    head->prev = newNode;
    head = newNode;

    int position = 4;
    newNode = new Node;
    newNode->data = 'E';

    if (position == 1) {
        newNode->next = head;
        newNode->prev = NULL;
        head->prev = newNode;
        head = newNode;
    } else {
        Node *temp = head;
        for (int i = 2; i < position;
i++) {
            if (temp->next != NULL) {
                temp = temp->next;
            } else {
                cout << "Previous node
cannot be null." << endl;
                delete newNode;
                return 1;
            }
        }
        newNode->next = temp->next;
        if (temp->next != NULL) {
            temp->next->prev = newNode;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }

    traverseList(head);

     return 0;
}
```

C/C++

```
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
    Node *prev;
};

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;
    head->prev = NULL;
```

- In the node there is a prev pointer added
- The <-> displays the lists double linked list

```
    second->data = 'P';
    second->next = third;
    second->prev = head;

    third->data = 'E';
    third->next = fourth;
    third->prev = second;

    fourth->data = '0';
    fourth->next = fifth;
    fourth->prev = third;

    fifth->data = '1';
    fifth->next = last;
    fifth->prev = fourth;

    last->data = '0';
    last->next = NULL;
    last->prev = fifth;

    Node *newNode = new Node;
    newNode->data = '1';
    newNode->next = NULL;
    newNode->prev = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        Node *temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }

    Node *temp = head;
    printf("\n\nThe elements in the list
are: - \n");
    while (temp != NULL) {
        printf("%c", temp->data);
        temp = temp->next;
        if (temp != NULL) {
            printf(" <-> ");
        }
    }

    temp = head;
    while (temp != NULL) {
        Node *nextNode = temp->next;
        delete temp;
        temp = nextNode;
    }

    return 0;
```

```
}
```

```
C/C++
#include <iostream>
using namespace std;

class Node {
public:
    char data;
    Node *next;
    Node *prev;
};

int main() {
    Node *head = NULL;
    Node *second = NULL;
    Node *third = NULL;
    Node *fourth = NULL;
    Node *fifth = NULL;
    Node *last = NULL;

    head = new Node;
    second = new Node;
    third = new Node;
    fourth = new Node;
    fifth = new Node;
    last = new Node;

    head->data = 'C';
    head->next = second;
    head->prev = NULL;

    second->data = 'P';
    second->next = third;
    second->prev = head;

    third->data = 'E';
    third->next = fourth;
    third->prev = second;

    fourth->data = '0';
    fourth->next = fifth;
    fourth->prev = third;

    fifth->data = '1';
    fifth->next = last;
    fifth->prev = fourth;
```

- In the node there is a prev pointer added
- The <-> displays the lists double linked list

```
    last->data = '0';
    last->next = NULL;
    last->prev = fifth;

    int position = 3;
    Node *temp = head;

    if (position == 1) {
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        delete temp;
    } else {
        for (int i = 2; i < position;
i++) {
            temp = temp->next;
        }
        Node *nodeToDelete = temp->next;
        if (nodeToDelete != NULL) {
            temp->next =
nodeToDelete->next;
            if (nodeToDelete->next !=
NULL) {
                nodeToDelete->next->prev
= temp;
            }
            delete nodeToDelete;
        }
    }

    Node *printTemp = head;
    printf("\n\nThe elements in the list
are: - \n");
    while (printTemp != NULL) {
        printf("%c", printTemp->data);
        printTemp = printTemp->next;
        if (printTemp != NULL) {
            printf(" <-> ");
        }
    }

    return 0;
}
```

7. Supplementary Activity

C/C++

```cpp
#include <iostream>
#include <string>
using namespace std;

class Song {
public:
    string title;

    Song(string t) : title(t) {}
};

class Node {
public:
    Song* data;
    Node* next;
    Node* prev;

    Node(Song* song) : data(song), next(nullptr), prev(nullptr) {}
};

class CircularDoublyLinkedList {
private:
    Node* head;
    Node* tail;
    int size;

public:
    CircularDoublyLinkedList() : head(nullptr), tail(nullptr), size(0) {}

    void append(Song* song) {
        Node* newNode = new Node(song);
        if (size == 0) {
            head = newNode;
            tail = newNode;
            head->next = head;
            head->prev = head;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            newNode->next = head;
            head->prev = newNode;
            tail = newNode;
        }
        size++;
    }

    void remove(string title) {
        if (size == 0) return;

        Node* current = head;
        do {
            if (current->data->title == title) {
                if (current == head) head = current->next;
                if (current == tail) tail = current->prev;

                current->prev->next = current->next;
                current->next->prev = current->prev;

                delete current->data;
                delete current;
                size--;
                return;
            }
            current = current->next;
        } while (current != head);
    }

    void traverse() {
        if (size == 0) {
            cout << "The playlist is empty." << endl;
            return;
        }

        Node* current = head;
        cout << "Playlist: ";
        do {
            cout << current->data->title << " <-> ";
            current = current->next;
        } while (current != head);
        cout << "(back to start)" << endl;
    }

    void playAll() {
        if (size == 0) {
            cout << "The playlist is empty." << endl;
            return;
        }

        Node* current = head;
        cout << "Playing all songs:" << endl;
        do {
            cout << "Now playing: " << current->data->title << endl;
            current = current->next;
        } while (current != head);
    }

    ~CircularDoublyLinkedList() {
        while (size > 0) {
            remove(head->data->title);
        }
    }
};

int main() {
    CircularDoublyLinkedList playlist;

    Song* song1 = new Song("Espresso");
    Song* song2 = new Song("Shape Of You");
    Song* song3 = new Song("Macarena");
    Song* song4 = new Song("Closer");

    playlist.append(song1);
    playlist.append(song2);
    playlist.append(song3);
    playlist.append(song4);

    playlist.traverse();
    playlist.playAll();

    cout << "Removing 'Shape Of You' from the playlist." << endl;
    playlist.remove("Shape Of You");
    playlist.traverse();
    playlist.playAll();

    return 0;
}
```

8. Conclusion

I've learned about the various applications of linked lists, such as single linked lists with nodes that contain a data element and a pointer to the next node, double linked lists with traversal in both directions, and circular linked lists with looping through the list. Manage song playlists and learn about node insertion, deletion, and traversal. This activity helped me gain a better understanding of linked lists. I believe I could perform better in this activity. I need to practice and improve my skills so that I can apply what I've learned.

9. Assessment Rubric