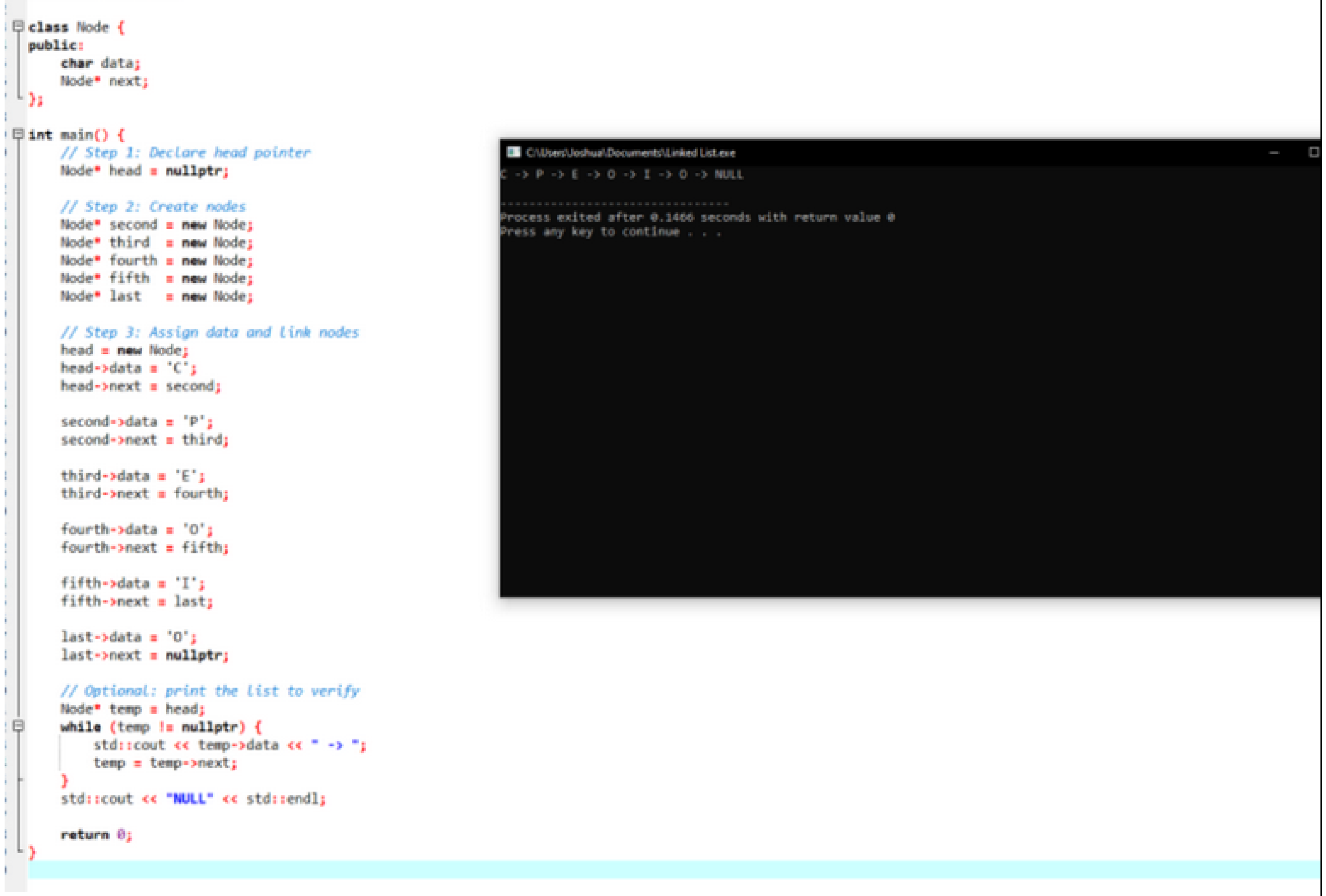
Activity No. <n>	
<Replace with Title>	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 08/14/25**
Section: CPE 010-CPE21S4	**Date Submitted: 08/14/25**
Name(s):Catungal Kerwin Jan B	**Instructor: Jimlord Quejado**

6. Output



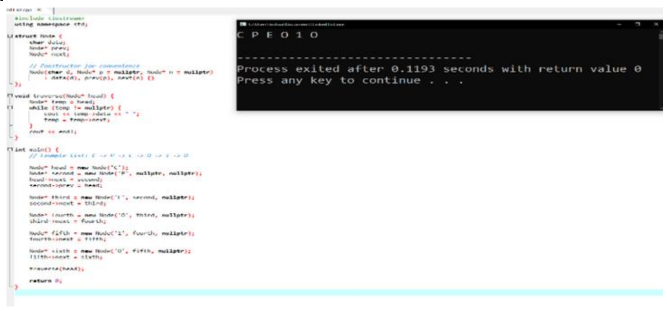
```
class Node {
public:
    char data;
    Node* next;
};

int main() {
    // Step 1: Declare head pointer
    Node* head = nullptr;

    // Step 2: Create nodes
    Node* second = new Node;
    Node* third  = new Node;
    Node* fourth = new Node;
    Node* fifth  = new Node;
    Node* last   = new Node;

    // Step 3: Assign data and link nodes
    head = new Node;
    head->data = 'C';
    head->next = second;

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

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

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

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

    last->data = 'O';
    last->next = nullptr;

    // Optional: print the list to verify
    Node* temp = head;
    while (temp != nullptr) {
        std::cout << temp->data << " -> ";
        temp = temp->next;
    }
    std::cout << "NULL" << std::endl;

    return 0;
}
```

C:\Users\Joshua\Documents\Linked List.exe

```
C -> P -> E -> O -> I -> O -> NULL

--------------------------------
Process exited after 0.1466 seconds with return value 0
Press any key to continue . . .
```

Discussion: Here I implemented linked list where I can store characters and display it on a sequence order.

Operation:

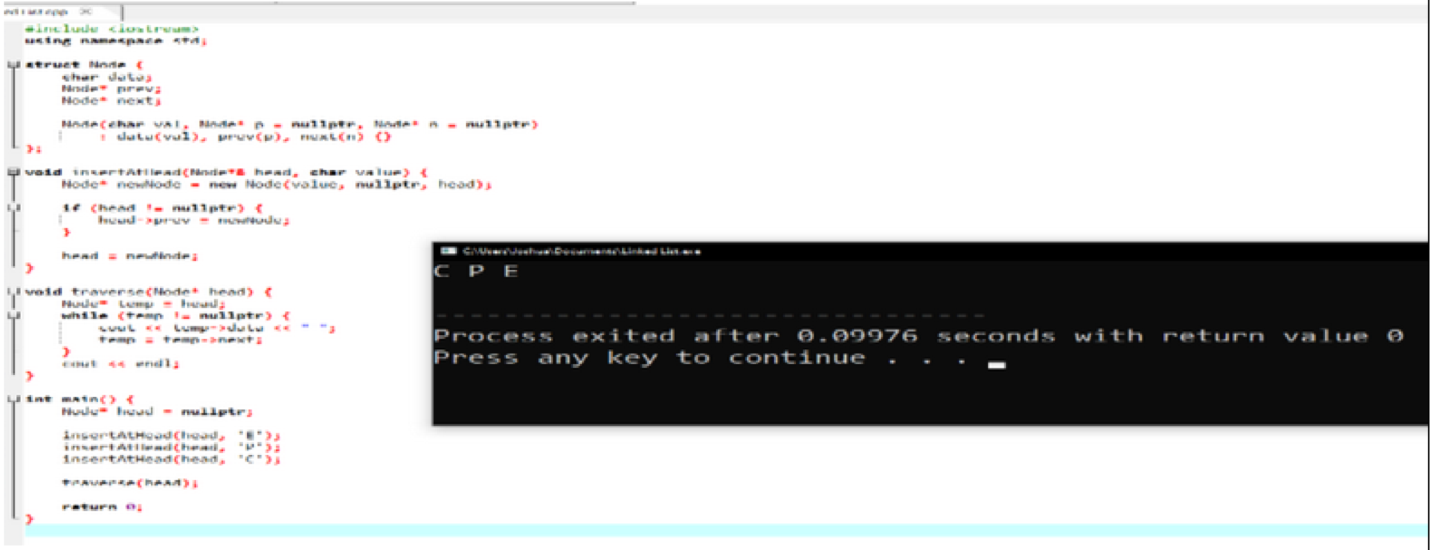


```
#include <iostream>
using namespace std;

struct Node {
    char data;
    Node* prev;
    Node* next;

    // Constructor for convenience
    Node(char d, Node* p = nullptr, Node* n = nullptr)
        : data(d), prev(p), next(n) {}
};

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    // Example list: C -> P -> E -> O -> 1 -> O

    Node* head = new Node('C');
    Node* second = new Node('P', nullptr, nullptr);
    head->next = second;
    second->prev = head;

    Node* third = new Node('E', second, nullptr);
    second->next = third;

    Node* fourth = new Node('O', third, nullptr);
    third->next = fourth;

    Node* fifth = new Node('1', fourth, nullptr);
    fourth->next = fifth;

    Node* sixth = new Node('O', fifth, nullptr);
    fifth->next = sixth;

    traverse(head);

    return 0;
}
```

```
C P E O 1 O

--------------------------------
Process exited after 0.1193 seconds with return value 0
Press any key to continue . . .
```

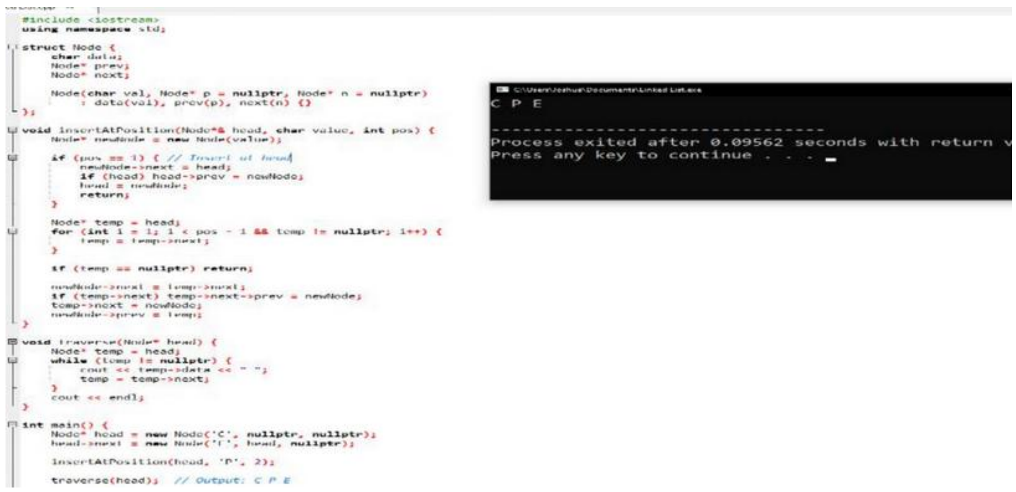
Traversal:

Insertion at Head:



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

struct Node {
    char data;
    Node* prev;
    Node* next;

    Node(char val, Node* p = nullptr, Node* n = nullptr)
        : data(val), prev(p), next(n) {}
};

void insertAtHead(Node*& head, char value) {
    Node* newNode = new Node(value, nullptr, head);

    if (head != nullptr) {
        head->prev = newNode;
    }

    head = newNode;
}

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;

    insertAtHead(head, 'E');
    insertAtHead(head, 'P');
    insertAtHead(head, 'C');

    traverse(head);

    return 0;
}
```

```
C P E

--------------------------------
Process exited after 0.09976 seconds with return value 0
Press any key to continue . . . _
```

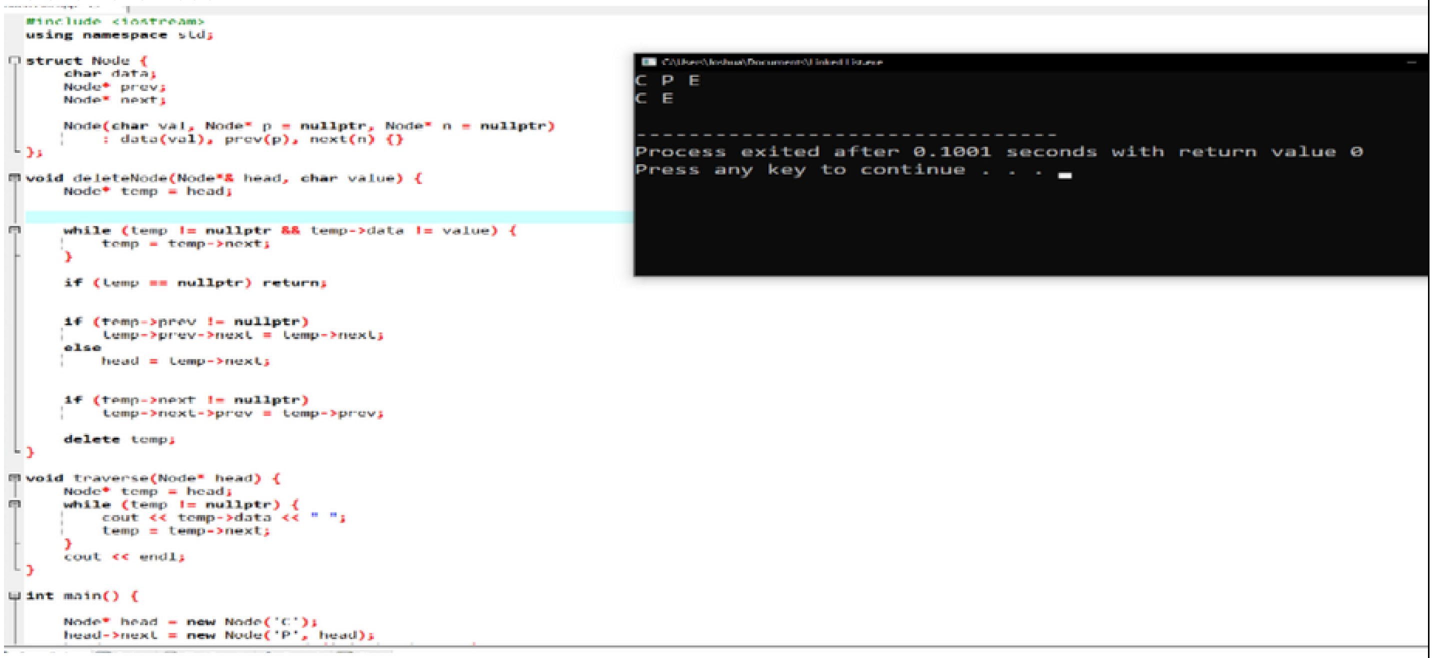


```cpp
#include <iostream>
using namespace std;

struct Node {
    char data;
    Node* prev;
    Node* next;

    Node(char val, Node* p = nullptr, Node* n = nullptr)
        : data(val), prev(p), next(n) {}
};

void insertAtPosition(Node*& head, char value, int pos) {
    Node* newNode = new Node(value);

    if (pos == 1) { // Insert at head
        newNode->next = head;
        if (head) head->prev = newNode;
        head = newNode;
        return;
    }

    Node* temp = head;
    for (int i = 1; i < pos - 1 && temp != nullptr; i++) {
        temp = temp->next;
    }

    if (temp == nullptr) return;

    newNode->next = temp->next;
    if (temp->next) temp->next->prev = newNode;
    temp->next = newNode;
    newNode->prev = temp;
}

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = new Node('C', nullptr, nullptr);
    head->next = new Node('E', head, nullptr);

    insertAtPosition(head, 'P', 2);

    traverse(head);  // Output: C P E
```

```
C P E

--------------------------------
Process exited after 0.09562 seconds with return v
Press any key to continue . . . _
```

Insertion at the end:

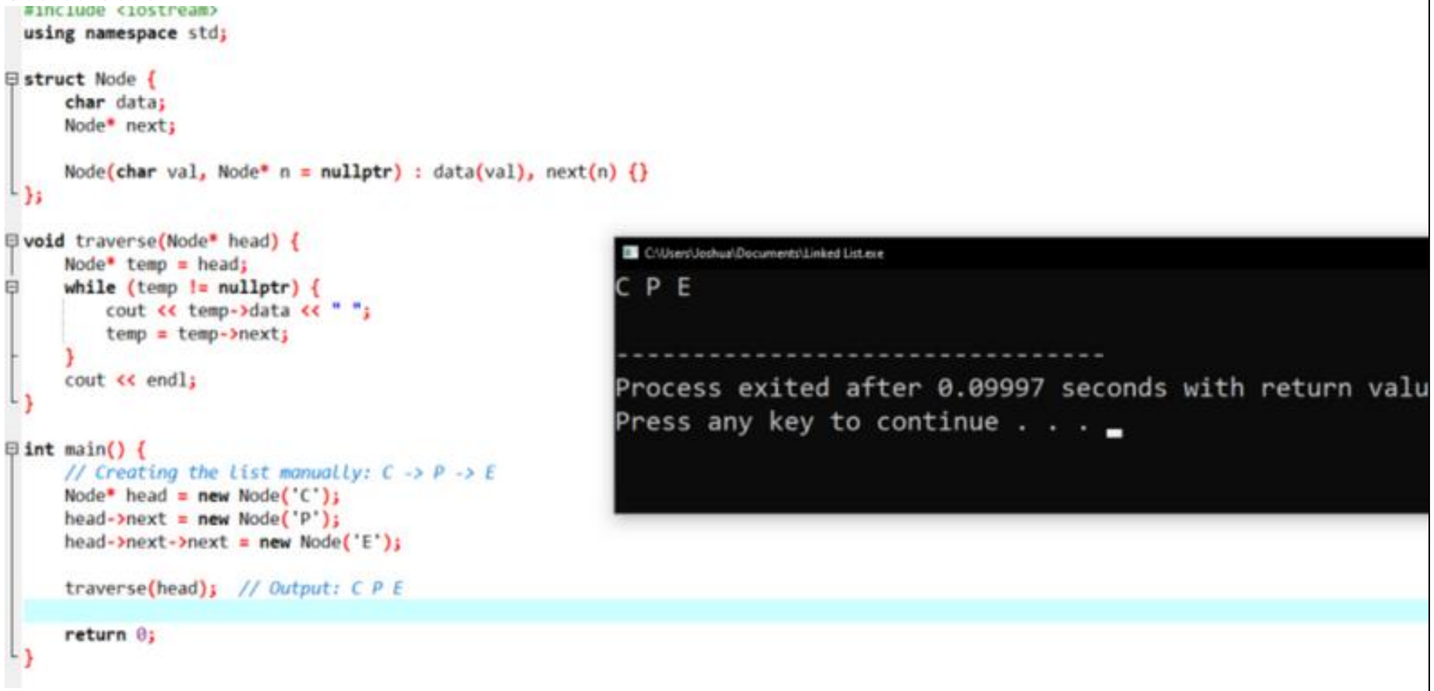
Deletion of a node:



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

struct Node {
    char data;
    Node* prev;
    Node* next;

    Node(char val, Node* p = nullptr, Node* n = nullptr)
        : data(val), prev(p), next(n) {}
};

void deleteNode(Node*& head, char value) {
    Node* temp = head;

    while (temp != nullptr && temp->data != value) {
        temp = temp->next;
    }

    if (temp == nullptr) return;


    if (temp->prev != nullptr)
        temp->prev->next = temp->next;
    else
        head = temp->next;


    if (temp->next != nullptr)
        temp->next->prev = temp->prev;

    delete temp;
}

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {

    Node* head = new Node('C');
    head->next = new Node('P', head);
```

```
C P E
C E

--------------------------------
Process exited after 0.1001 seconds with return value 0
Press any key to continue . . . _
```

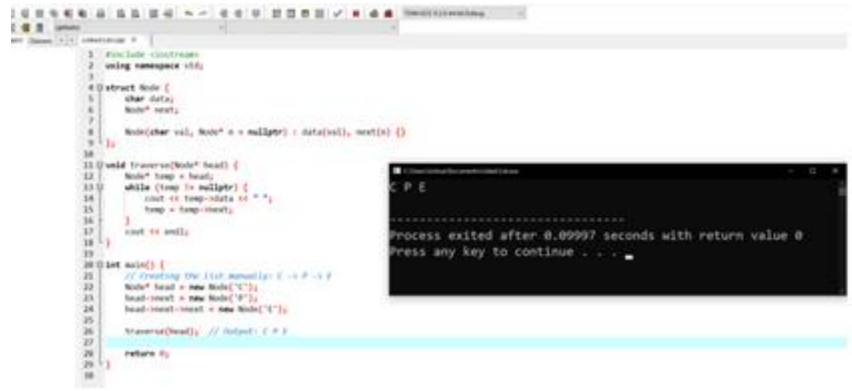
Source Code:



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

struct Node {
    char data;
    Node* next;

    Node(char val, Node* n = nullptr) : data(val), next(n) {}
};

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    // Creating the list manually: C -> P -> E
    Node* head = new Node('C');
    head->next = new Node('P');
    head->next->next = new Node('E');

    traverse(head);  // Output: C P E

    return 0;
}
```

```
C:\Users\Joshua\Documents\Linked List.exe
C P E

--------------------------------
Process exited after 0.09997 seconds with return valu
Press any key to continue . . .
```

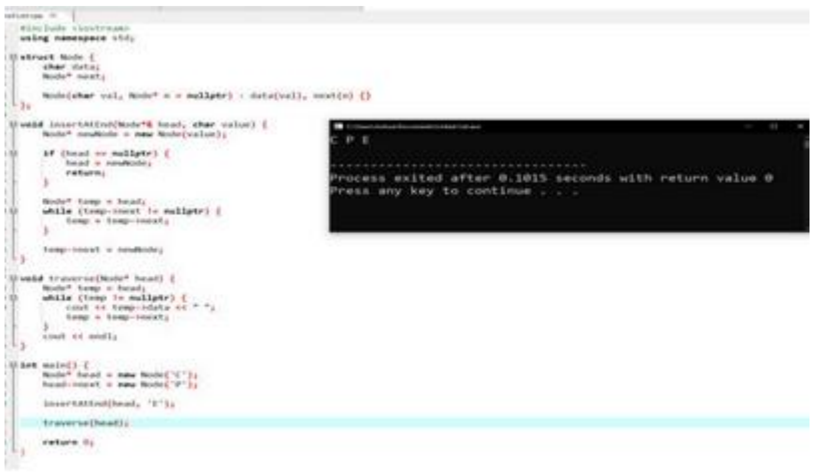
Source code:



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

struct Node {
    char data;
    Node* next;

    Node(char val, Node* n = nullptr) : data(val), next(n) {}
};

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    // Creating the list manually: C -> P -> E
    Node* head = new Node('C');
    head->next = new Node('P');
    head->next->next = new Node('E');

    traverse(head);  // Output: C P E

    return 0;
}
```

```
C P E

--------------------------------
Process exited after 0.09997 seconds with return value 0
Press any key to continue . . .
```

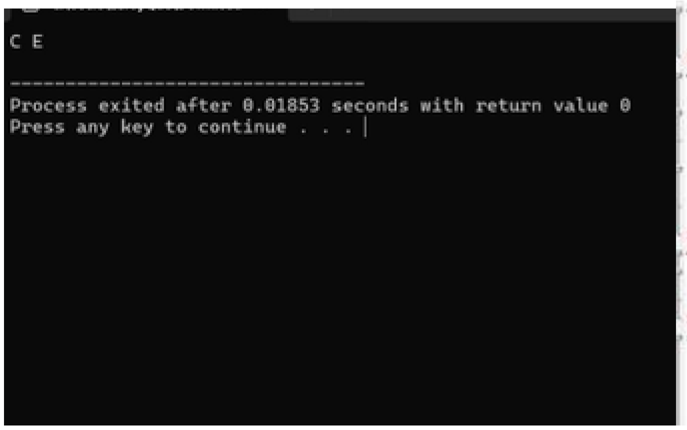
Source code:



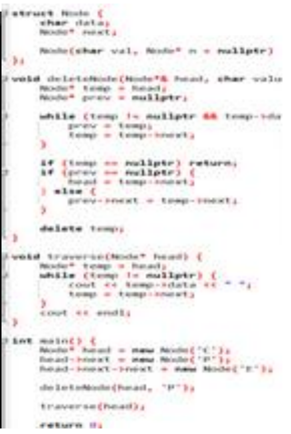
```cpp
#include <iostream>
using namespace std;

struct Node {
    char data;
    Node* next;

    Node(char val, Node* n = nullptr) : data(val), next(n) {}
};

void insertAtEnd(Node*& head, char value) {
    Node* newNode = new Node(value);

    if (head == nullptr) {
        head = newNode;
        return;
    }

    Node* temp = head;
    while (temp->next != nullptr) {
        temp = temp->next;
    }

    temp->next = newNode;
}

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = new Node('C');
    head->next = new Node('P');

    insertAtEnd(head, 'E');

    traverse(head);

    return 0;
}
```

```
C P E

--------------------------------
Process exited after 0.1015 seconds with return value 0
Press any key to continue . . .
```

Source Code:

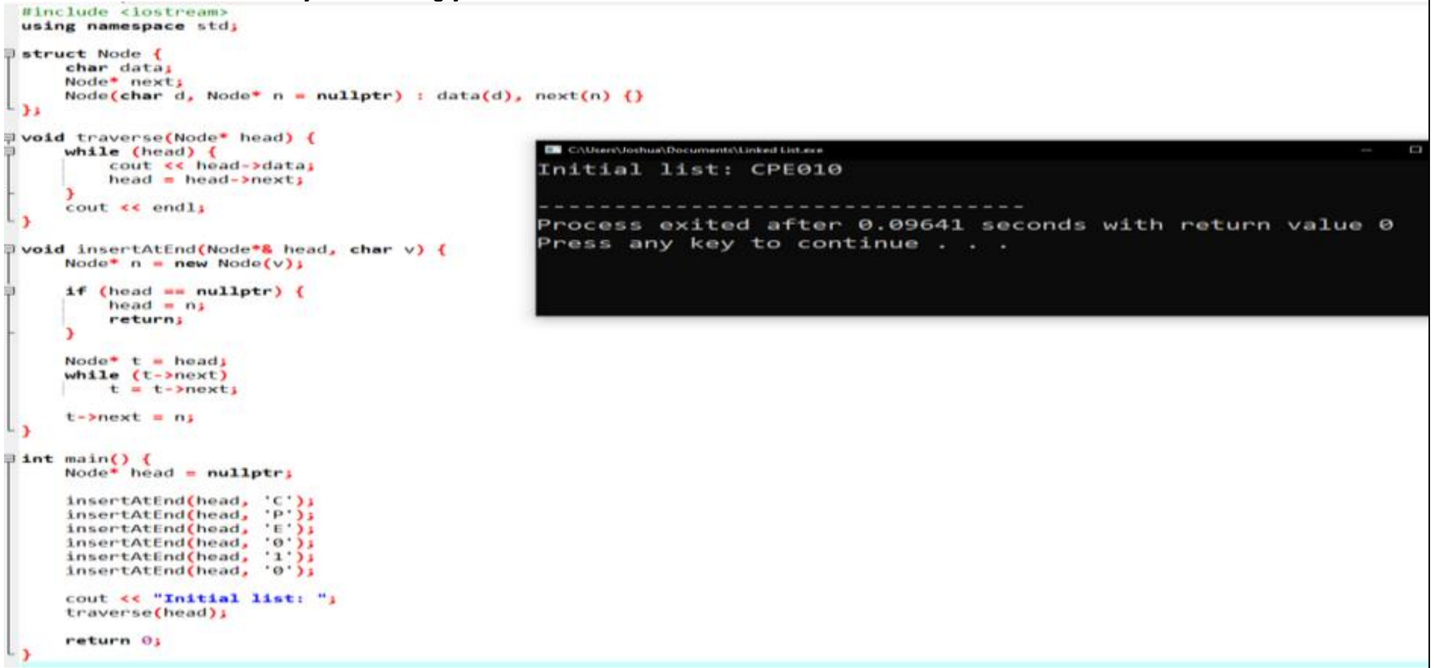


```
C E

--------------------------------
Process exited after 0.01853 seconds with return value 0
Press any key to continue . . .
```



```
struct Node {
    char data;
    Node* next;

    Node(char val, Node* n = nullptr)
};

void deleteNode(Node*& head, char valu
    Node* temp = head;
    Node* prev = nullptr;

    while (temp != nullptr && temp->da
        prev = temp;
        temp = temp->next;
    }

    if (temp == nullptr) return;
    if (prev == nullptr) {
        head = temp->next;
    } else {
        prev->next = temp->next;
    }

    delete temp;
}

void traverse(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = new Node('C');
    head->next = new Node('P');
    head->next->next = new Node('E');

    deleteNode(head, 'P');

    traverse(head);

    return 0;
```

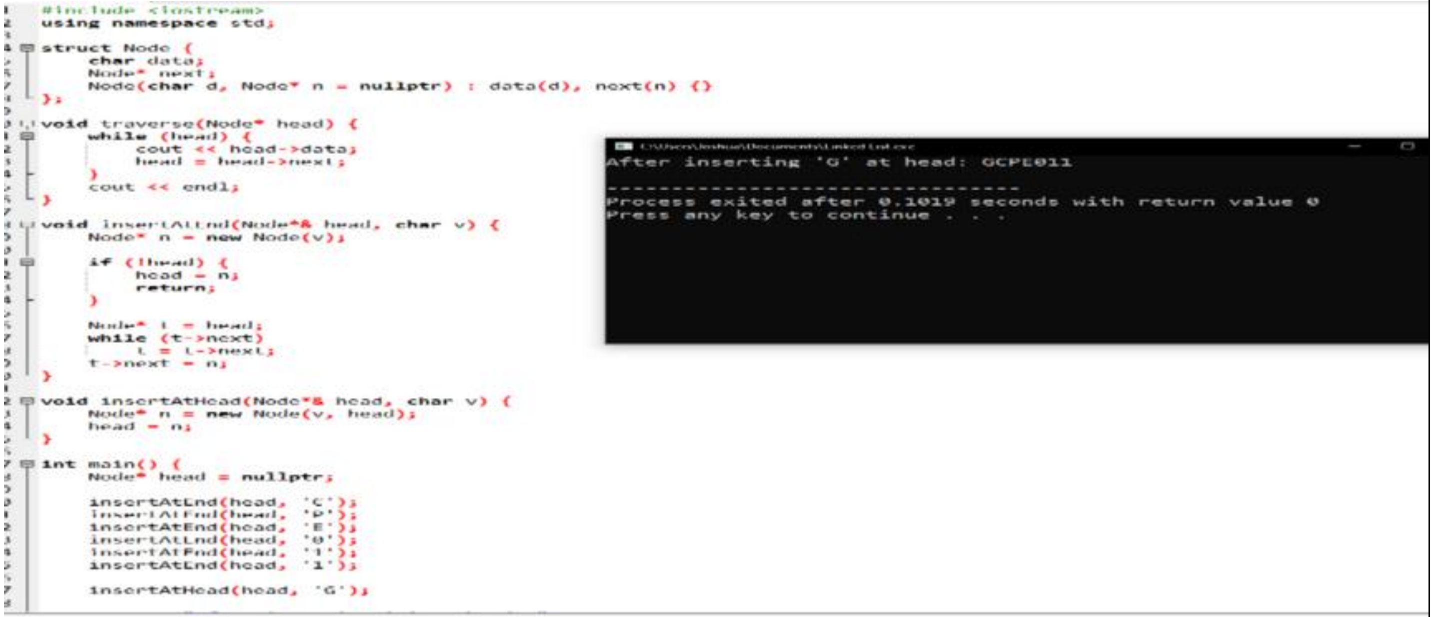
Table 3-3. Code and Analysis for Singly Linked Lists



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

struct Node {
    char data;
    Node* next;
    Node(char d, Node* n = nullptr) : data(d), next(n) {}
};

void traverse(Node* head) {
    while (head) {
        cout << head->data;
        head = head->next;
    }
    cout << endl;
}

void insertAtEnd(Node*& head, char v) {
    Node* n = new Node(v);

    if (head == nullptr) {
        head = n;
        return;
    }

    Node* t = head;
    while (t->next)
        t = t->next;

    t->next = n;
}

int main() {
    Node* head = nullptr;

    insertAtEnd(head, 'C');
    insertAtEnd(head, 'P');
    insertAtEnd(head, 'E');
    insertAtEnd(head, '0');
    insertAtEnd(head, '1');
    insertAtEnd(head, '0');

    cout << "Initial list: ";
    traverse(head);

    return 0;
}
```

```
C:\Users\Joshua\Documents\Linked List.exe
Initial list: CPE010

--------------------------------
Process exited after 0.09641 seconds with return value 0
Press any key to continue . . .
```

Analysis:
By traversing the list by making the head pointer pass. Here the Function walks from node to node and prints the stored characters.



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

struct Node {
    char data;
    Node* next;
    Node(char d, Node* n = nullptr) : data(d), next(n) {}
};

void traverse(Node* head) {
    while (head) {
        cout << head->data;
        head = head->next;
    }
    cout << endl;
}

void insertAtEnd(Node*& head, char v) {
    Node* n = new Node(v);

    if (!head) {
        head = n;
        return;
    }

    Node* t = head;
    while (t->next)
        t = t->next;
    t->next = n;
}

void insertAtHead(Node*& head, char v) {
    Node* n = new Node(v, head);
    head = n;
}

int main() {
    Node* head = nullptr;

    insertAtEnd(head, 'C');
    insertAtEnd(head, 'P');
    insertAtEnd(head, 'E');
    insertAtEnd(head, '0');
    insertAtEnd(head, '1');
    insertAtEnd(head, '1');

    insertAtHead(head, 'G');
```

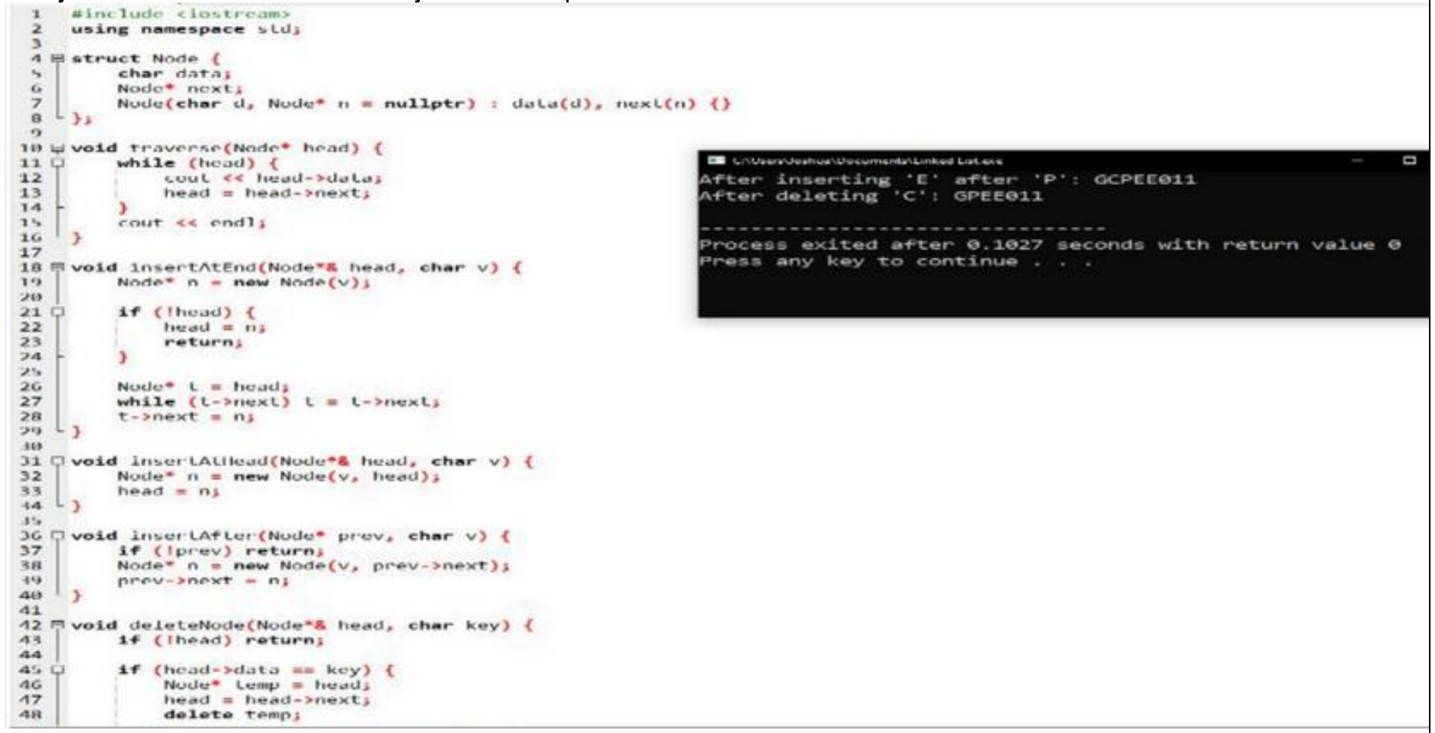
```
C:\Users\Joshua\Documents\Linked List.exe
After inserting 'G' at head: GCPE011

--------------------------------
Process exited after 0.1019 seconds with return value 0
Press any key to continue . . .
```

Analysis: here it creates a new node wherein next points to the previous head, then reassigns head.

```
clude <iostream>
ing namespace std;

ruct Node {
  char data;
  Node* next;
  Node(char d, Node* n = nullptr) : data(d), next(n) {}

id traverse(Node* head) {
  while (head) {
      cout << head->data;
      head = head->next;
  }
  cout << endl;

id insertAtEnd(Node*& head, char v) {
  Node* n = new Node(v);

  if (!head) {
      head = n;
      return;
  }

  Node* t = head;
  while (t->next)
      t = t->next;
  t->next = n;

id insertAtHead(Node*& head, char v) {
  Node* n = new Node(v, head);
  head = n;

id insertAfter(Node* prev, char v) {
  if (!prev) return;
  Node* n = new Node(v, prev->next);
  prev->next = n;

t main() {
  Node* head = nullptr;

  insertAtEnd(head, '[illegible]');
  insertAtEnd(head, '[illegible]');
  insertAtEnd(head, '[illegible]');
```

```
After inserting '[illegible]' after '[illegible]': [illegible]
--------------------------------
Process exited after [illegible] seconds with return value 0
Press any key to continue . . .
```

Analysis: allocate new node and adjust the next pointers.

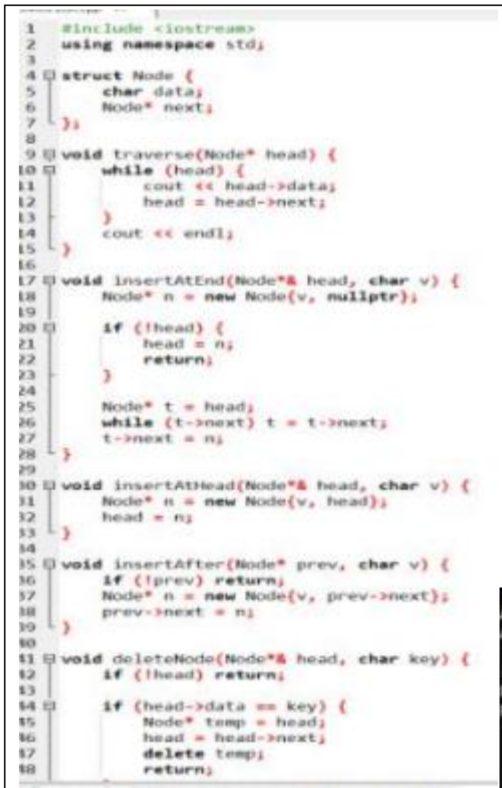


```
#include <iostream>
using namespace std;

struct Node {
    char data;
    Node* next;
    Node(char d, Node* n = nullptr) : data(d), next(n) {}
};

void traverse(Node* head) {
    while (head) {
        cout << head->data;
        head = head->next;
    }
    cout << endl;
}

void insertAtEnd(Node*& head, char v) {
    Node* n = new Node(v);

    if (!head) {
        head = n;
        return;
    }

    Node* t = head;
    while (t->next) t = t->next;
    t->next = n;
}

void insertAtHead(Node*& head, char v) {
    Node* n = new Node(v, head);
    head = n;
}

void insertAfter(Node* prev, char v) {
    if (!prev) return;
    Node* n = new Node(v, prev->next);
    prev->next = n;
}

void deleteNode(Node*& head, char key) {
    if (!head) return;

    if (head->data == key) {
        Node* temp = head;
        head = head->next;
        delete temp;
```

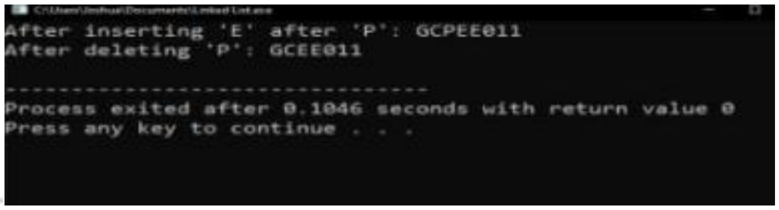
C:\Users\Joshua\Documents\Linked List.exe

```
After inserting 'E' after 'P': GCPEE011
After deleting 'C': GPEE011

--------------------------------
Process exited after 0.1027 seconds with return value 0
Press any key to continue . . .
```

Analysis: Deleting a node by locating the node before the desired target.

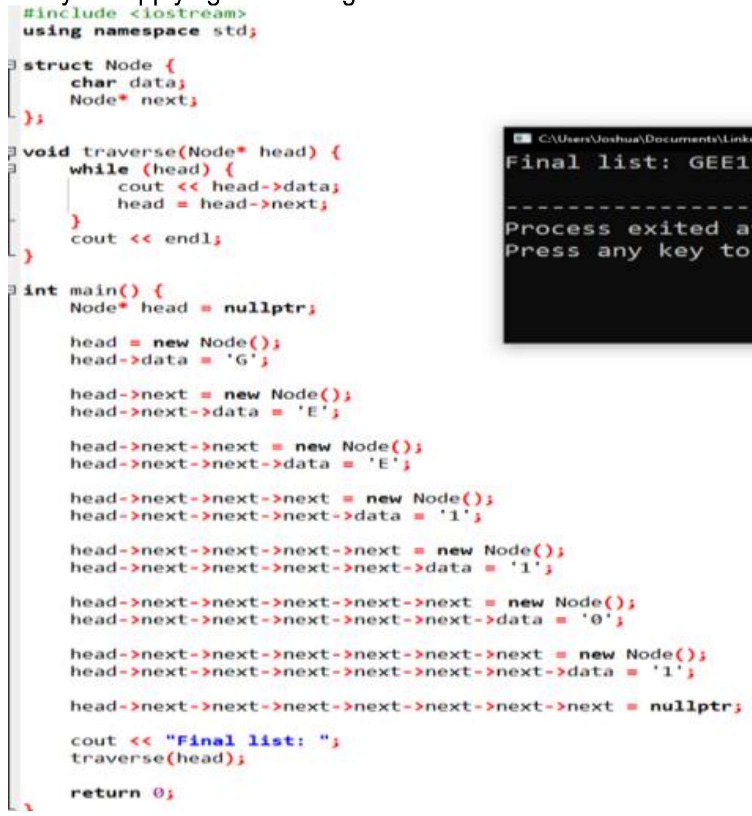


```cpp
#include <iostream>
using namespace std;

struct Node {
    char data;
    Node* next;
};

void traverse(Node* head) {
    while (head) {
        cout << head->data;
        head = head->next;
    }
    cout << endl;
}

void insertAtEnd(Node*& head, char v) {
    Node* n = new Node{v, nullptr};

    if (!head) {
        head = n;
        return;
    }

    Node* t = head;
    while (t->next) t = t->next;
    t->next = n;
}

void insertAtHead(Node*& head, char v) {
    Node* n = new Node{v, head};
    head = n;
}

void insertAfter(Node* prev, char v) {
    if (!prev) return;
    Node* n = new Node{v, prev->next};
    prev->next = n;
}

void deleteNode(Node*& head, char key) {
    if (!head) return;

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

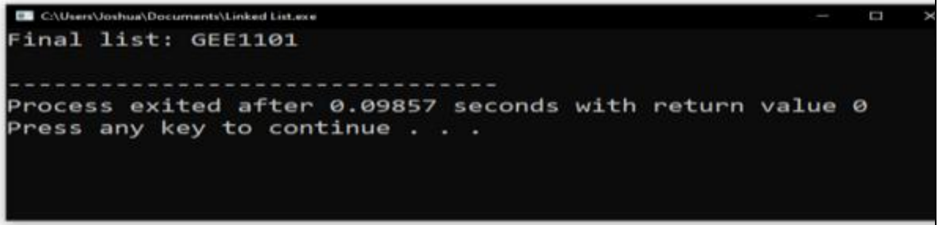


```
After inserting 'E' after 'P': GCPEE011
After deleting 'P': GCEE011

--------------------------------
Process exited after 0.1046 seconds with return value 0
Press any key to continue . . .
```

Analysis: Applying deletion again to remove "P".



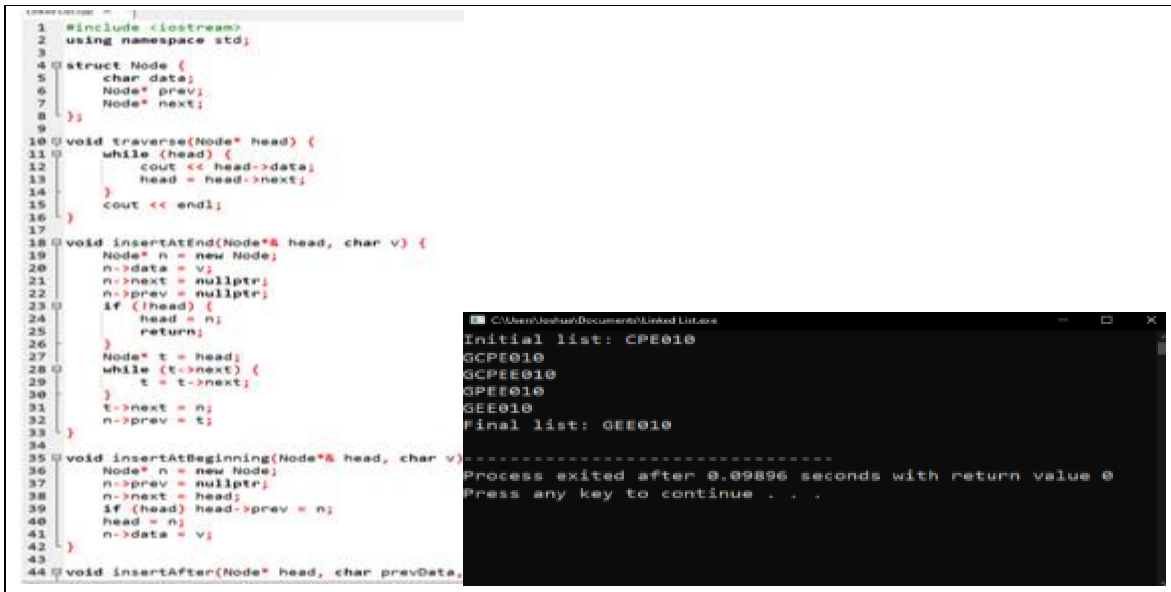
```cpp
#include <iostream>
using namespace std;

struct Node {
    char data;
    Node* next;
};

void traverse(Node* head) {
    while (head) {
        cout << head->data;
        head = head->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;

    head = new Node();
    head->data = 'G';

    head->next = new Node();
    head->next->data = 'E';

    head->next->next = new Node();
    head->next->next->data = 'E';

    head->next->next->next = new Node();
    head->next->next->next->data = '1';

    head->next->next->next->next = new Node();
    head->next->next->next->next->data = '1';

    head->next->next->next->next->next = new Node();
    head->next->next->next->next->next->data = '0';

    head->next->next->next->next->next->next = new Node();
    head->next->next->next->next->next->next->data = '1';

    head->next->next->next->next->next->next->next = nullptr;

    cout << "Final list: ";
    traverse(head);

    return 0;
}
```



```
Final list: GEE1101

--------------------------------
Process exited after 0.09857 seconds with return value 0
Press any key to continue . . .
```

Analysis: result after all operations.

Table 3-4. Modified Operations for Doubly Linked Lists

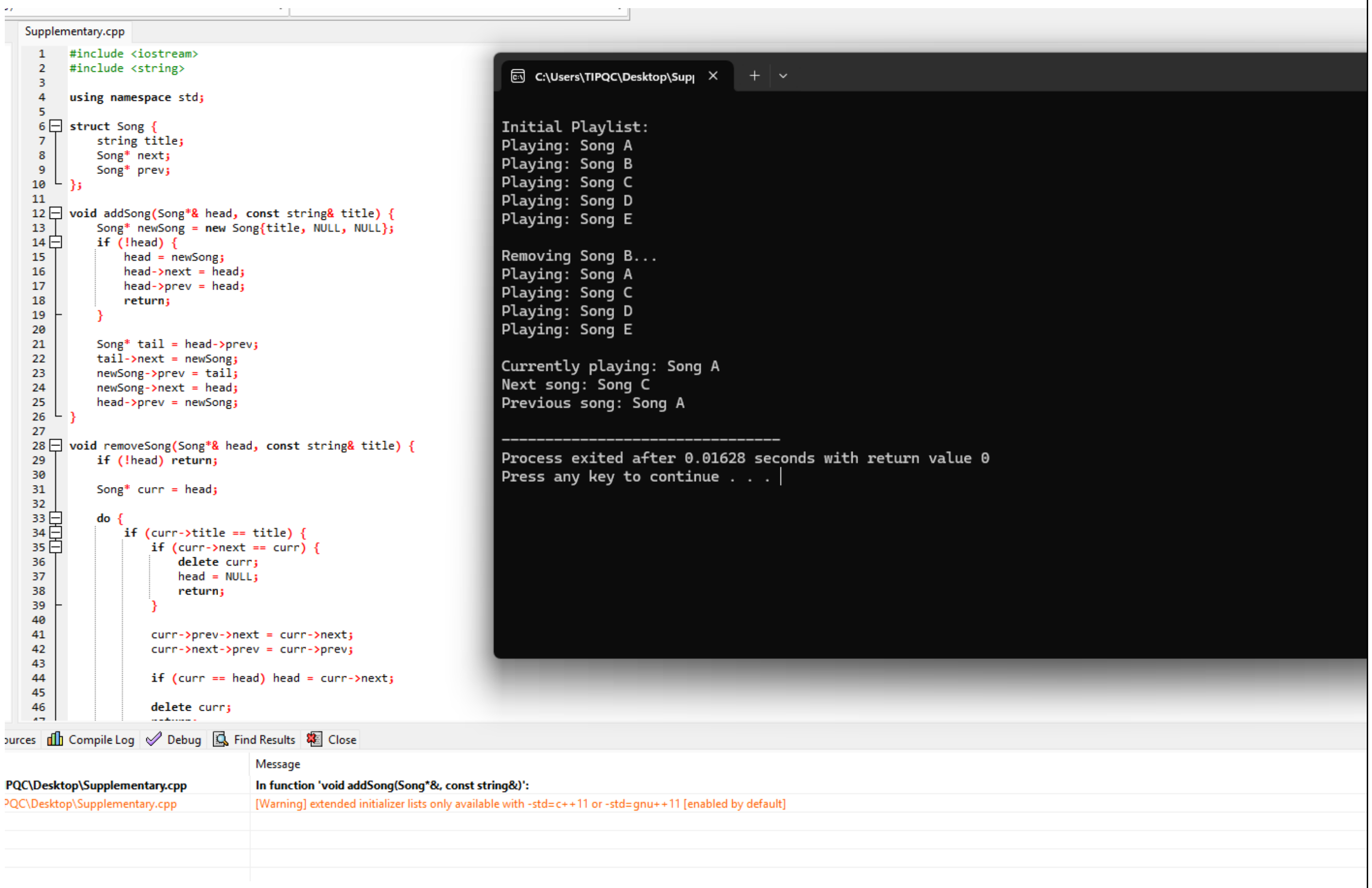


```cpp
#include <iostream>
using namespace std;

struct Node {
    char data;
    Node* prev;
    Node* next;
};

void traverse(Node* head) {
    while (head) {
        cout << head->data;
        head = head->next;
    }
    cout << endl;
}

void insertAtEnd(Node*& head, char v) {
    Node* n = new Node;
    n->data = v;
    n->next = nullptr;
    n->prev = nullptr;
    if (!head) {
        head = n;
        return;
    }
    Node* t = head;
    while (t->next) {
        t = t->next;
    }
    t->next = n;
    n->prev = t;
}

void insertAtBeginning(Node*& head, char v)
    Node* n = new Node;
    n->prev = nullptr;
    n->next = head;
    if (head) head->prev = n;
    head = n;
    n->data = v;
}

void insertAfter(Node* head, char prevData,
```

C:\Users\Joshua\Documents\Linked List.exe

```
Initial list: CPE010
GCPE010
GCPEE010
GPEE010
GEE010
Final list: GEE010

--------------------------------
Process exited after 0.09896 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

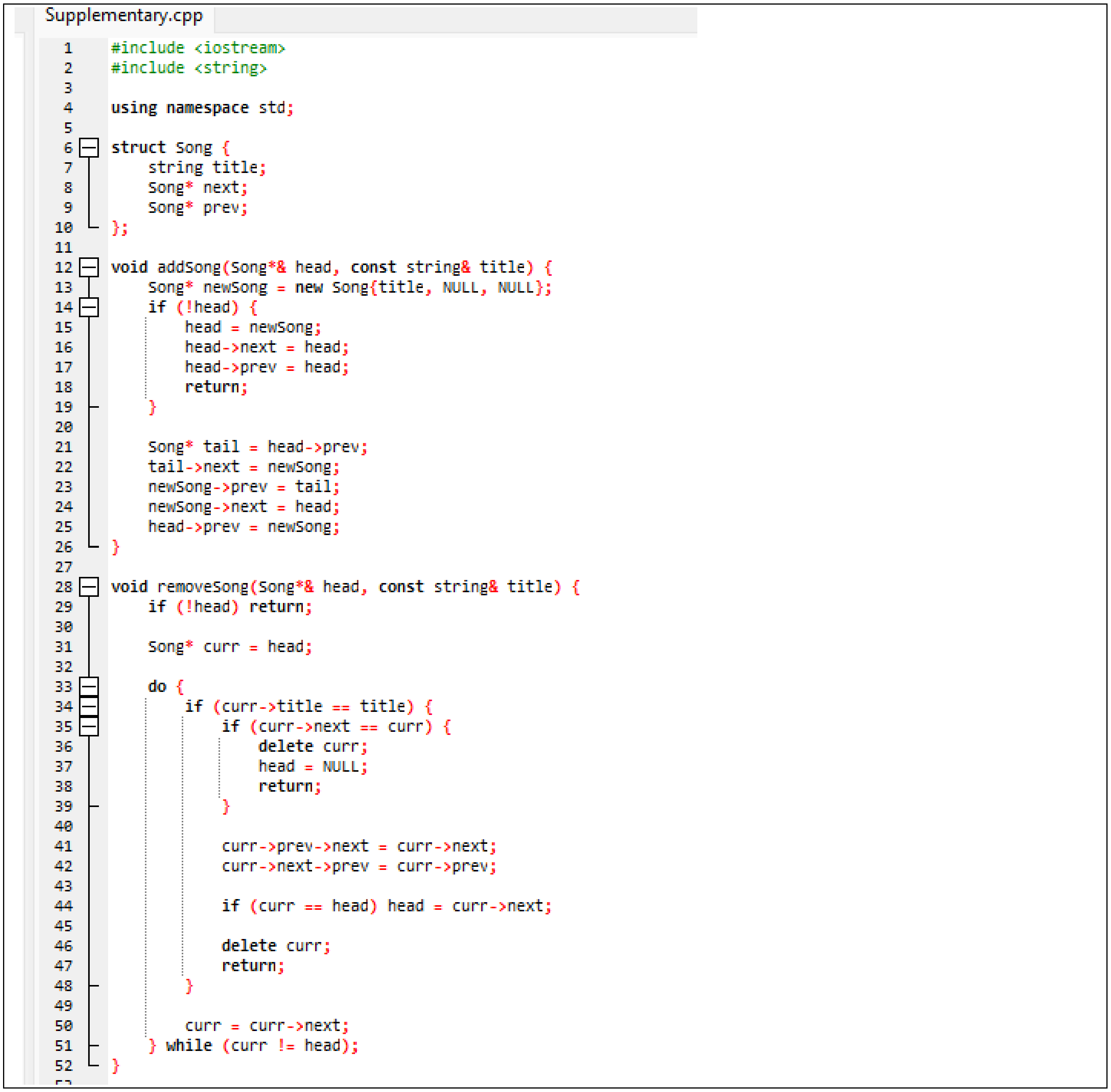


```cpp
#include <iostream>
#include <string>

using namespace std;

struct Song {
    string title;
    Song* next;
    Song* prev;
};

void addSong(Song*& head, const string& title) {
    Song* newSong = new Song{title, NULL, NULL};
    if (!head) {
        head = newSong;
        head->next = head;
        head->prev = head;
        return;
    }

    Song* tail = head->prev;
    tail->next = newSong;
    newSong->prev = tail;
    newSong->next = head;
    head->prev = newSong;
}

void removeSong(Song*& head, const string& title) {
    if (!head) return;

    Song* curr = head;

    do {
        if (curr->title == title) {
            if (curr->next == curr) {
                delete curr;
                head = NULL;
                return;
            }

            curr->prev->next = curr->next;
            curr->next->prev = curr->prev;

            if (curr == head) head = curr->next;

            delete curr;
```

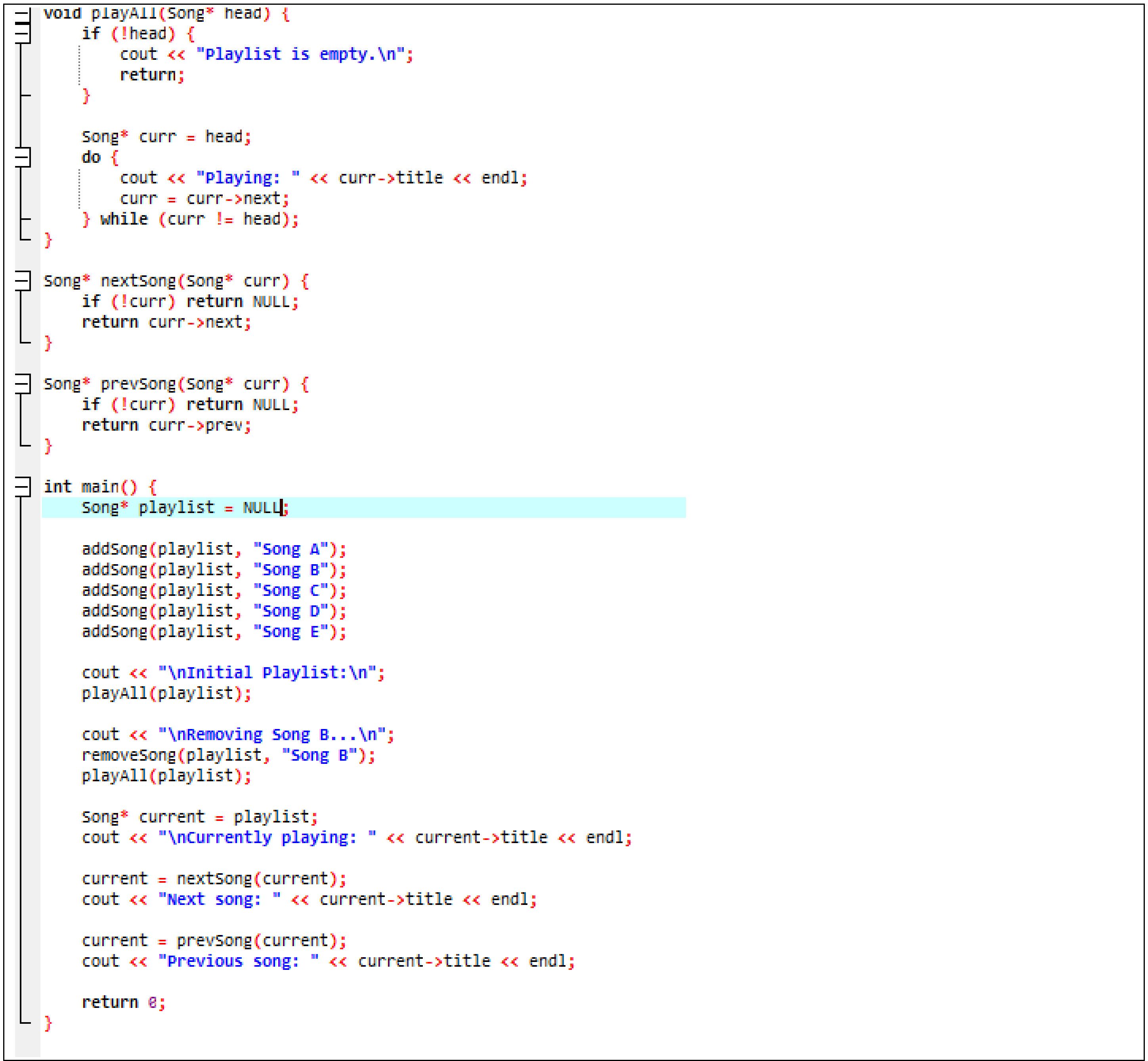
C:\Users\TIPQC\Desktop\Sup

```
Initial Playlist:
Playing: Song A
Playing: Song B
Playing: Song C
Playing: Song D
Playing: Song E

Removing Song B...
Playing: Song A
Playing: Song C
Playing: Song D
Playing: Song E

Currently playing: Song A
Next song: Song C
Previous song: Song A

--------------------------------
Process exited after 0.01628 seconds with return value 0
Press any key to continue . . .
```

```
PQC\Desktop\Supplementary.cpp    In function 'void addSong(Song*&, const string&)':
PQC\Desktop\Supplementary.cpp    [Warning] extended initializer lists only available with -std=c++11 or -std=gnu++11 [enabled by default]
```


Supplementary.cpp

```cpp
#include <iostream>
#include <string>

using namespace std;

struct Song {
    string title;
    Song* next;
    Song* prev;
};

void addSong(Song*& head, const string& title) {
    Song* newSong = new Song{title, NULL, NULL};
    if (!head) {
        head = newSong;
        head->next = head;
        head->prev = head;
        return;
    }

    Song* tail = head->prev;
    tail->next = newSong;
    newSong->prev = tail;
    newSong->next = head;
    head->prev = newSong;
}

void removeSong(Song*& head, const string& title) {
    if (!head) return;

    Song* curr = head;

    do {
        if (curr->title == title) {
            if (curr->next == curr) {
                delete curr;
                head = NULL;
                return;
            }

            curr->prev->next = curr->next;
            curr->next->prev = curr->prev;

            if (curr == head) head = curr->next;

            delete curr;
            return;
        }

        curr = curr->next;
    } while (curr != head);
}
```



```
void playAll(Song* head) {
    if (!head) {
        cout << "Playlist is empty.\n";
        return;
    }

    Song* curr = head;
    do {
        cout << "Playing: " << curr->title << endl;
        curr = curr->next;
    } while (curr != head);
}

Song* nextSong(Song* curr) {
    if (!curr) return NULL;
    return curr->next;
}

Song* prevSong(Song* curr) {
    if (!curr) return NULL;
    return curr->prev;
}

int main() {
    Song* playlist = NULL;

    addSong(playlist, "Song A");
    addSong(playlist, "Song B");
    addSong(playlist, "Song C");
    addSong(playlist, "Song D");
    addSong(playlist, "Song E");

    cout << "\nInitial Playlist:\n";
    playAll(playlist);

    cout << "\nRemoving Song B...\n";
    removeSong(playlist, "Song B");
    playAll(playlist);

    Song* current = playlist;
    cout << "\nCurrently playing: " << current->title << endl;

    current = nextSong(current);
    cout << "Next song: " << current->title << endl;

    current = prevSong(current);
    cout << "Previous song: " << current->title << endl;

    return 0;
}
```

8. Conclusion

This activity allow and teach us how to modify singly linked lists, it helped me understand how pointers work especially the process of carefully updating next and prev pointers.

9. Assessment Rubric