

```

1  #include <iostream>
2  using namespace std;
3
4  template <typename T>
5  class Node {
6  public:
7      T data;
8      Node* next;
9
10     Node(T data) : data(data), next(nullptr) {}
11 };
12
13 template <typename T>
14 class LinkedList {
15 private:
16     Node<T>* head;
17     int size;
18
19 public:
20     LinkedList() : head(nullptr), size(0) {}
21
22     // Insert at the beginning
23     void insertAtBeginning(T data) {
24         Node<T>* newNode = new Node<T>(data);
25         newNode->next = head;
26         head = newNode;
27         size++;
28     }
29
30     // Insert at the end
31     void insertAtEnd(T data) {
32         Node<T>* newNode = new Node<T>(data);
33         if (head == nullptr) {
34             head = newNode;
35         } else {
36             Node<T>* current = head;
37             while (current->next != nullptr) {
38                 current = current->next;
39             }
40             current->next = newNode;
41         }
42         size++;
43     }
44
45     // Delete at the beginning
46     void deleteAtBeginning() {
47         if (head == nullptr) {
48             return;
49         }
50         Node<T>* temp = head;
51         head = head->next;
52         delete temp;
53         size--;
54     }
55
56     // Print the linked list
57     void printList() {
58         Node<T>* current = head;
59         while (current != nullptr) {
60             cout << current->data << " ";
61             current = current->next;
62         }
63         cout << endl;
64     }
65
66     // Get the size of the linked list

```

```

67     int getSize() {
68         return size;
69     }
70
71     // Destructor to free memory
72     ~LinkedList() {
73         Node<T>* current = head;
74         while (current != nullptr) {
75             Node<T>* temp = current;
76             current = current->next;
77             delete temp;
78         }
79         head = nullptr;
80     }
81 };
82
83 int main() {
84     LinkedList<int> intList;
85     intList.insertAtEnd(10);
86     intList.insertAtBeginning(5);
87     intList.insertAtEnd(15);
88
89     cout << "Linked List: ";
90     intList.printList(); // Output: Linked List: 5 10 15
91
92     cout << "Size of List: " << intList.getSize() << endl; // Output: Size of List: 3
93
94     intList.deleteAtBeginning();
95
96     cout << "Deleting the element at the first of the list" << endl;
97
98     cout << "Linked List after deletion: ";
99     intList.printList(); // Output: Linked List: 10 15
100
101     return 0;
102 }

```