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

//creating the node

struct Node{
	int data;
	Node *next; //it is Node type pointer as it points the next node.
};

class Singly{
	private:
		Node *start;
	public:
		Singly(){
			start = NULL;
		}

		//insert at the end
		void insertAtEnd(int val){

			Node *newNode = new Node;
			newNode->data = val;
			newNode->next = NULL;
//			check if the node linked list is empty or not
			if(start == NULL){
				start = newNode;
			}
			else {
				Node *currentNode = start;
				while(currentNode->next != NULL){
				currentNode = currentNode->next;
				}
				currentNode->next = newNode;
			}
		}

		//delete at the end
		void deleteAtEnd(){

			Node *temp1 = start, *temp2;
			while(temp1->next!=NULL){
				temp2 = temp1;
				temp1= temp1->next;
```

```
                }
                temp2->next = NULL;
                delete temp1;

        }

//      function to insert at start
        void insertAtStart(int val){
//              creating the node
                Node *newNode = new Node;
                newNode->data = val;
                newNode->next = NULL;

                newNode->next = start;
                start = newNode;

        }

//      function to delete at start
        void deleteAtStart(){
                Node *temp = new Node;
                temp=start;
                start = temp->next;
                delete temp;


        }
//      insert at given index
        void insertAtIndex(int val, int index) {
           Node *newNode = new Node;
           newNode->data = val;
           newNode->next = NULL;

           int i = 1;
           Node *currentNode = start;

           if (index == 0) {
              newNode->next = start;
              start = newNode;
           } else {
              while (index > i) {
                 if (currentNode->next != NULL) {
                    currentNode = currentNode->next;
                    ++i;
```

```
            } else {
                cout << "Index is very large so, inserting at end" << endl;
                break;
            }
        }

        newNode->next = currentNode->next;
        currentNode->next = newNode;
    }
}


//          delete at specific index
            void deleteAtIndex(int index){
                if (index < 0) {
                cout<< "Invalid index" << endl;
                return;
            }

            if (start == NULL) {
                cout << "List is empty" << endl;
                return;
            }

            if (index == 0) {
                Node* temp = start;
                start = start->next;
                delete temp;
                return;
            }

            Node* current = start;
            int currentIndex = 0;

            while (current != NULL && currentIndex < index - 1) {
                current = current->next;
                currentIndex++;
            }

            if (current == NULL || current->next == NULL) {
                cout << "Index out of bounds" <<endl;
                return;
            }
```

```
                Node* temp = current->next;
                current->next = temp->next;
                delete temp;


            }


            // Function to display the contents of the list
        void display() {
            Node* current = start;
            while (current != NULL) {
                cout << current->data << " ";
                current = current->next;
            }
            cout << endl;
        }
};

int main(){
        Singly sing;

        sing.insertAtEnd(5);
        sing.insertAtEnd(6);
        sing.insertAtEnd(7);
        sing.insertAtStart(4);
        sing.insertAtIndex(10,0);
//      sing.deleteAtStart();
//      sing.deleteAtEnd();
        sing.deleteAtIndex(3);
        sing.display();
}
```