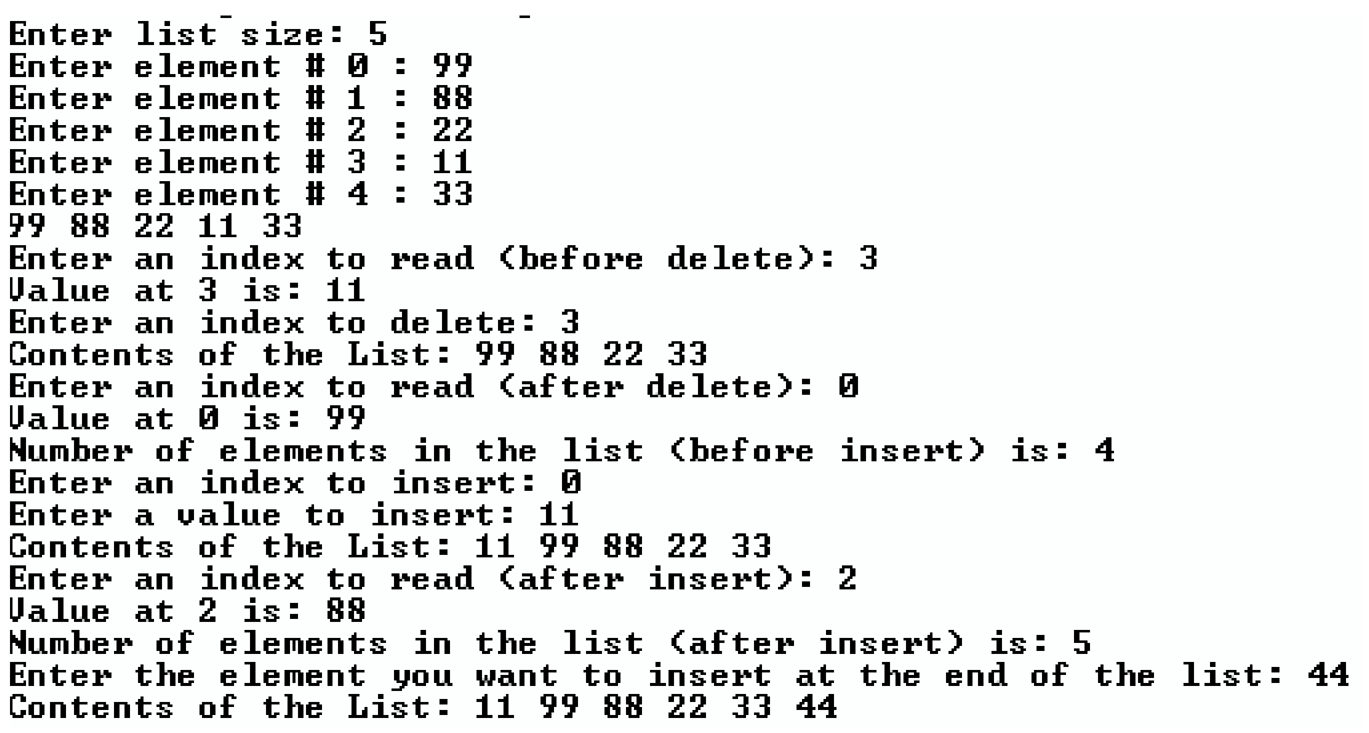
```cpp
#include <iostream>
using namespace std;

// implementing the dynamic List ADT using array
// operations to be implemented: read, Modify, delete, isEmpty, insert, countElements

class List{

    private:
        int *array;
        int maxSize; // useful to decide if resizing (doubling the array size) is needed
        int endOfArray;

    public:
        List(int size){
            array = new int[size];
            maxSize = size;
            endOfArray = -1;
        }

        bool isEmpty(){

            if (endOfArray == -1)
                return true;

            return false;
        }

        void resize(int s){

            int *tempArray = array;

            array = new int[s];

            for (int index = 0; index < min(s, endOfArray+1); index++){
                array[index] = tempArray[index];
            }

            maxSize = s;

        }


        void insert(int data){

            if (endOfArray == maxSize-1)
                resize(2*maxSize);

            array[++endOfArray] = data;

        }


        void insertAtIndex(int insertIndex, int data){

            // if the user enters an invalid insertIndex, the element is
            // appended to the array, after the current last element
            if (insertIndex > endOfArray+1)
                insertIndex = endOfArray+1;

            if (endOfArray == maxSize-1)
                resize(2*maxSize);

            for (int index = endOfArray; index >= insertIndex; index--)
```

```
                array[index+1] = array[index];

            array[insertIndex] = data;
            endOfArray++;

        }


        int read(int index){
            return array[index];
        }

        void modifyElement(int index, int data){
            array[index] = data;
        }


        void deleteElement(int deleteIndex){

            // shift elements one cell to the left starting from deleteIndex+1 to endOfArray-1
            // i.e., move element at deleteIndex + 1 to deleteIndex and so on

            for (int index = deleteIndex; index < endOfArray; index++)
                array[index] = array[index+1];

            endOfArray--;

        }

        int countList(){
            int count = 0;
            for (int index = 0; index <= endOfArray; index++)
                count++;

            return count;
        }

        void print(){

            for (int index = 0; index <= endOfArray; index++)
                cout << array[index] << " ";

            cout << endl;

        }

};

int main(){

    int listSize;

    cout << "Enter list size: ";
    cin >> listSize;

    List integerList(1); // we will set the maxSize to 1 and double it as and when needed

    for (int i = 0; i < listSize; i++){

        int value;
        cout << "Enter element # " << i << " : ";
        cin >> value;

```

```cpp
            integerList.insertAtIndex(i, value);
        }

        integerList.print();


        // to read an element at a particular index (before delete)

        int readIndex;
        cout << "Enter an index to read (before delete): ";
        cin >> readIndex;
        cout << "Value at " << readIndex << " is: " << integerList.read(readIndex) << endl;

        // to delete an element at a particular index

        int deleteIndex;
        cout << "Enter an index to delete: ";
        cin >> deleteIndex;
        integerList.deleteElement(deleteIndex);

        cout << "Contents of the List: ";
        integerList.print();


        // to read an element at a particular index (after delete)

        cout << "Enter an index to read (after delete): ";
        cin >> readIndex;
        cout << "Value at " << readIndex << " is: " << integerList.read(readIndex) << endl;

        cout << "Number of elements in the list (before insert) is: " << integerList.countList() << endl;


        // to insert an element at a particular index
        int insertIndex, insertValue;
        cout << "Enter an index to insert: ";
        cin >> insertIndex;
        cout << "Enter a value to insert: ";
        cin >> insertValue;
        integerList.insertAtIndex(insertIndex, insertValue);

        cout << "Contents of the List: ";
        integerList.print();

        // to read an element at a particular index (after insert)

        cout << "Enter an index to read (after insert): ";
        cin >> readIndex;
        cout << "Value at " << readIndex << " is: " << integerList.read(readIndex) << endl;

        cout << "Number of elements in the list (after insert) is: " << integerList.countList() << endl;

        // to insert at the end of the list
        cout << "Enter the element you want to insert at the end of the list: ";
        cin >> insertValue;
        integerList.insert(insertValue);

        cout << "Contents of the List: ";
        integerList.print();


    return 0;
```

```
}
```



```
Enter list size: 5
Enter element # 0 : 99
Enter element # 1 : 88
Enter element # 2 : 22
Enter element # 3 : 11
Enter element # 4 : 33
99 88 22 11 33
Enter an index to read (before delete): 3
Value at 3 is: 11
Enter an index to delete: 3
Contents of the List: 99 88 22 33
Enter an index to read (after delete): 0
Value at 0 is: 99
Number of elements in the list (before insert) is: 4
Enter an index to insert: 0
Enter a value to insert: 11
Contents of the List: 11 99 88 22 33
Enter an index to read (after insert): 2
Value at 2 is: 88
Number of elements in the list (after insert) is: 5
Enter the element you want to insert at the end of the list: 44
Contents of the List: 11 99 88 22 33 44
```