**Report:**

1. **My Sequence was implemented using a doubly linked list, with the head pointing to a sentinel node. The nodes consist of a pointer to the previous node ( a nullptr if it is the first node on the list) and a pointer to the next node ( a nullptr if it is the last node on the list). For an empty list the head points to a nullptr.**
2. **Pseudocode:**

**int subsequence(const Sequence& seq1, const Sequence& seq2)**

**{**

**if seq2 is empty return 0;**

**if seq2 is larger than seq1 return -1;**

**iterate through seq1**

**iterate through seq2**

**to find matches;**

**if match found increment else break;**

**return the pos if reached end of seq2;**

**else return -1 match not found;**

**}**

**void interleave(const Sequence& seq1, const Sequence& seq2, Sequence& result)**

**{**

**if either sequences are empty set the other as result**

**if both are equal then insert alternately values of seq1 before values of seq2 into result**

**if seq1 < seq2 insert alternately till end of seq1 then insert the rest of seq2 into result**

**if seq2 < seq3 insert alternately till end of seq2 then insert the rest of seq3 into result**

**}**

**bool Sequence::insert(int pos, const ItemType& value)**

**{**

**if the pos is out of bounds return false;**

**check if pos is at the beginning of the list and list is not empty**

**insert value accordingly**

**check if list is empty**

**insert value accordingly**

**check if pos is the end of the list**

**insert accordingly**

**else (pos is somewhere in the middle)**

**insert accordingly**

**increment size and return true;**

**}**

**bool Sequence::erase(int pos)**

**{**

**if the position is out of bounds or the list is empty**

**return false;**

**iterate through the list till pos is found**

**{**

**if pos is the last on the list**

**if only one item on the list**

**delete accordingly;**

**else**

**delete accordingly;**

**else if pos is the first item on the list**

**delete accordingly;**

**else if pos is in the middle of the list**

**delete accordingly;**

**increment counter and change temp node to stay in the loop;**

**}**

**}**

1. **Test Cases:**

**ItemType m = 123;**

**Sequence s;**

**assert(s.size() == 0); //tests size of empty sequence**

**assert(s.empty() == true); //tests empty for empty sequence**

**assert(s.insert(0, 5)); //tests insert for empty sequence**

**assert(s.insert(0, 4)); //tests insert for beginning of sequence**

**assert(s.insert(2, 7)); //tests insert for end of sequence**

**assert(s.insert(2, 6)); //tests insert for middle of sequence**

**assert(s.size() == 4); //makes sure size works in insert function**

**assert(s.get(2, m) && m == 6); //makes sure s has the right order**

**Sequence d(s); //d should hold the same stuff as s**

**assert(d.erase(0) == true); //tests erase for beginning of sequence**

**assert(d.get(0, m) && m == 5); //makes sure erase worked and tests get function**

**assert(d.erase(1) == true); //tests erase for middle of sequence**

**assert(d.erase(1) == true); //tests erase for end of sequence**

**assert(d.get(0, m) && m == 5); //makes sure everything is in the right order**

**assert(d.size() == 1); //makes sure size works in erase function**

**Sequence t(s);**

**assert(t.size() == 4); //tests copy constructor**

**Sequence r;**

**r.insert(0);**

**r.insert(1);**

**r.insert(2);**

**r.insert(3);**

**r.insert(1,6);**

**assert(r.size()==5);**

**ItemType v;**

**r.get(2, v);**

**assert(v==1);**

**Sequence u;**

**interleave(r,s,u);**

**assert(u.size() == 9);**

**assert(u.get(0,v));**

**assert(v==0);**

**assert(u.get(5, v));**

**assert(v==6);**

**assert(u.remove(6) == 2); // tests remove function**

**assert(u.size() == 7);**

**assert(u.remove(0) == 1);**

**assert(u.remove(3) == 1);**

**// to check if set works**

**Sequence g;**

**for (int i = 0; i < 5; i++)**

**{**

**g.insert(0, i);**

**}**

**for (int i = 0; i < 5; i++)**

**{**

**g.set(i, i);**

**}**

**for (int i = 0; i < g.size(); i++)**

**{**

**g.get(i, v);**

**//cout << v << endl;**

**}**

**//to check if find works**

**assert(g.find(3) == 3);**

**assert(g.find(0) == 0);**

**assert(g.find(9) == -1);**

**assert(g.find(4) == 4);**

**//to check if swap works**

**Sequence sh;**

**Sequence a;**

**assert(sh.empty()); // check to see if s set is empty**

**assert(a.empty()); // check to see if a set is empty**

**for (int i = 0; i < 200; i++)**

**{**

**sh.insert(i); // Insert 200 items in s**

**}**

**assert(sh.size() == 200); // Check to see if s contains 200**

**sh.swap(a); // Swap the random values from a for numbers 0-200 into a**

**assert(a.size() == 200); // Check to see if the size is 200**

**assert(a.get(40, v) && v == 40);**

**assert(sh.size() == 0);**

**// checking subsequence**

**Sequence c;**

**Sequence b;**

**for(int i = 0; i < 3; i++)**

**c.insert(i, i+1);**

**assert(subsequence(g, c) == 1);**

**assert(subsequence(c, b) == 0);**

**assert(subsequence(c, g) == -1);**

**assert(subsequence(u, c) == -1);**