Taimur Rahman

Design

```
#include <iostream>


using namespace std;
template <typename Key,typename Info>
class Sequence{
private:
    struct element{
        Key key;
        Info info;
        element *next;
    };
    element *head;
    int number;
    void removeAllElements();
    void copyAllElements(const Sequence<Key,Info>& x);
public:
    Sequence();//constructor
    ~Sequence();//destructor
    Sequence(const Sequence<Key,Info>& x);//copy constructor
    Sequence<Key,Info>& operator=(const Sequence<Key,Info>& x);//assignment operator
    void insertElement(const Key& x, const Info& y);//insert to end of list
    element getHead();//return head
    void setHead(element* e);//head = e
    bool isEmpty() const;//checks if number = 0
    int numberOfElements()const;//return number
    void print(ostream& os) const;//print function
```

```
Sequence<Key,Info> Shuffle(const Sequence<Key,Info>&S1,int start1,int len1,const
Sequence<Key,Info>&S2,int start2, int len2, int num){
        //checking if anything is wrong parameter wise
        try{
            bool n = 0;
        if(len1<0||len2<0||start1>S1.number||start2>S2.number||num<0){
            throw n;
        }
        }catch(bool){
        cerr<<"Parameters are wrong";
        }
        //start
    Sequence<Key,Info> *result;
    result = new Sequence;
    int C = start1;//checks the current position for 1st sequence
    int D = start2;//checks the current position for 2nd sequence
    int cont = 0;//counter for both sequences
    element* xtr = S1.head;//element of first list
    element* ytr = S2.head;//element of second list
    while(cont!=start1){//checks when counter reaches the starting position
        xtr = xtr->next;//element increments
        cont++;//counter increments
    }//end of while loop
    cont = 0;
    while(cont!=start2){//checks when counter reaches the starting position
        ytr = ytr->next;//element increments
        cont++;//counter increments
    }//end of while loop
    for (int j=0;j<num;j++){//start of for loop for the third sequence
        for (int k=0;k<len1;k++){//nested for loop for the first sequence
           if (C<S1.number){//current element number less than entire list length
                result->insertElement(xtr->key,xtr->info);//inserts current position
                C++;//current position increments
                xtr = xtr->next;//element increments
           }//end of if statement
```

```
        }//end of nested for loop for first sequence
        for (int l=0;l<len2;l++){//nested for loop for the second sequence
            if (D<S2.number){//current element number less than entire list length
                result->insertElement(ytr->key,ytr->info);//inserts current position
                D++;//current position increments
                ytr = ytr->next;//element increments
            }//end of if statement
        }//end of nested for loop for second sequence
    }//end of for loop for third sequence
    return *result;
    };


};
```

Implementation

The Shuffle method uses two for loops nested inside a bigger for loop. At first, it finds the specific element that is needed to be added into the third list, by using a while loop to find the starting element from which the shuffling starts from. It is done for both sequences. In the two nested for loops, a counter (C and D) counts the current position and checks if it exceeds the length of the list.

The method does not work entirely with wrong parameters where it is either

1. Impossible (length cannot be negative)
2. Limited (starting position cannot be greater than length of the entire list)

If sum exceeds the number of elements able to be printed it stops printing for the list.