```
#include <iostream>

using namespace std;

class List {
  struct Node {
      char elem;
      Node* next;
      Node (char elem='_', Node* next=NULL) {
            this->elem=elem;
            this->next=next;
      }
  };
  Node* lst;
public:

  List () :lst(NULL) {}

  List (char c) {
        lst=new Node(c);
  }

  List (const char* s) {
        if (*s=='\0') {
            lst=NULL;
            return;
        }

        Node *p;
        lst= p= new Node(*s++);

        while (*s!='\0') {
            p= p->next= new Node(*s++);
        }
  }

  bool empty() const {return lst==NULL;}

    List (const List& l) {
        if (l.empty()) {
            lst=NULL;
            return;
        }
        Node *p, *pl=l.lst;
        p= lst= new Node(pl->elem);
        pl=pl->next;
        while (pl!=NULL) {
            p->next= new Node(pl->elem);
            p=p->next;
            pl=pl->next;
        }
    }

    void clear() {
        Node *pl;
        while (lst!=NULL) {
```

```
        pl=lst;
        lst=lst->next;
        delete pl;
    }
}

~List() {
    this->clear();
}


List operator+ (const List& l) {
    Node *p,*pl;

    if(this->empty()){
           return l;
    }

    List res(*this);

    p=res.lst;
    while(p->next!=NULL){
        p=p->next;
    }

    pl=l.lst;

    while (pl!=NULL) {
        p=p->next=new Node(pl->elem);
        pl=pl->next;
    }

    return res;
}

friend ostream& operator<< (ostream& s, const List& l) {
    if (l.lst==NULL) {
        s<<"";
        return s;
    }
    Node *pl=l.lst;
    while (pl!=NULL) {
        s<<pl->elem;
        pl=pl->next;

    }

    return s;
}


List& operator= (const List& l) {
    Node *p,*pl;

    if (&l==this){
            return *this;
    }
```

```
        this->clear();

        if (l.lst==NULL) {
            return *this;
        }
        p=lst= new Node(l.lst->elem);
        pl=l.lst->next;
        while (pl!=NULL) {
            p->next= new Node(pl->elem);
            p=p->next;
            pl=pl->next;

        }

        return *this;
    }
};

int main() {
    List l1,l2('a'), l3("abc"), l4(l1), l5(l3);
    cout<<l1<<' '<<l2<<' '<<l3<<' '<<l4<<' '<<l5<<endl;
    cout<<(l1=l4)<<' '<<(l3)<<' '<<(l4=l3)<<endl;
    cout<<l1+'a';
    cout<<l1-'a';
    cout<< 'a'-l1;// оставить в списке только буквы a Например было   abacaa ==> стало aaaa
    cout<<'a'+l1;


    return 0;
}
```