```
 1: #ifndef LISTA3_H2
 2: #define LISTA3_H2
 3:
 4: #include <iostream>
 5: using namespace std;
 6:
 7: struct node {
 8:   node(){};
 9:   node(const int val):data(val){};
10:   int data;
11:   node* next;
12:   node* prev;
13: };
14:
15: class lista {
16: public:
17:   lista() { head = new node; head->next = head; head->prev=head;}
18:   // copy constructor
19:   lista(const lista& listobj) {
20:     head = new node;
21:     head->next = head;
22:     head->prev = head;
23:     for (node* np = listobj.head->next; np != listobj.head; np = np->next) {
24:       push_back(np->data);
25:     }
26:   }
27:
28:   // destructor
29:   ~lista() {
30:     // fill in code to delete all nodes while traversing the list;
31:     // delete the head node;
32:   }
33:
34:   // assignment operator overload
35:   lista& operator = (const lista & _list)
36:   {
37:     head = new node;
38:     head->next = head;
39:     head->prev = head;
40:
41:     node *np;
42:     for (np = _list.head->next; np != _list.head; np = np->next)
43:       push_back(np->data);
44:
45:     return *this;
46:
47:   }
48:
49:   void push_front(const int& val) {
50:     node *np = new node(val);
51:     np->next = head->next;
52:     np->prev = head;
53:     head->next->prev = np;
54:     head->next = np;
55:   }
56:   void push_back(const int& val) {
57:     node *np = new node(val);
58:     np->prev = head->prev;
59:     np->next = head;
60:     head->prev->next = np;
61:     head->prev = np;
62:   }
63:   void display() {
64:     for (node* np = head->next; np != head; np=np->next)
```

```
65:       cout << np->data << endl;
66:   }
67: private:
68:   node* head;
69: };
70: #endif
```