```
#ifndef LIST_H
#define LIST_H

#include <iostream>

// forward declcarations
template <typename T> class List;

template <typename T>
class List {
        private:
        struct Node     // an all public class with data only, no member ftns
        {
                T       data;
                Node    *prev;
                Node    *next;

                Node(const T& d = T(), Node *p = NULL, Node *n = NULL) : \
                        data(d), prev(p), next(n) \
                                                { };
        };

        public:
        // constructors (overloaded)
        List();

        // copy constructor
        List(const List& rhs);

        // destructors
        ~List()                                 {
                                                  clear();
                                                  delete head;
                                                  delete tail;
                                                }

        // assignment operator
        const List&     operator=(const List&);

        // operators

        // iterator functions
        Node*   begin()                         { return head->next; }
        Node*   end()                           { return tail; }

        Node*           insert(Node*, const T&);
        void            erase(Node*);

        // members
        int             size() const            { return sz; }
        bool            empty() const           { return size() == 0; }
        void            clear()                 { while(!empty()) pop_front(); }

        T&              front()                 { return head->next->data; }
        const T&        front() const           { return head->next->data; }
        T&              back()                  { return tail->prev->data; }
        const T&        back() const            { return tail->prev->data; }
        Node*           push_front(const T& o)  { return insert(head->next, o); }
        Node*           push_back(const T& o)   { return insert(tail, o); }
        void            pop_front()             { erase(head->next); }
        void            pop_back()              { erase(tail); }

        // private: only available to this class
        private:
        int     sz;
        Node    *head;
        Node    *tail;
};

#endif
```