```
#ifndef LIST_H
#define LIST_H

#include <iostream>

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

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

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

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

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

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

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

        // operators

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

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

        // 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_t*         push_front(const T& o)  { return insert(head->next, o); }
        node_t*         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_t  *head;
        node_t  *tail;
};

#endif
```