```
class node
{
public:
    // TYPEDEF
    typedef double value_type;

    // CONSTRUCTOR
    node(
        const value_type& init_data = value_type( ),
        node* init_link = NULL
    )
    { data_field = init_data; link_field = init_link; }

    // Member functions to set the data and link fields:

    void set_data(const value_type& new_data) { data_field = new_data; }

    void set_link(node* new_link)             { link_field = new_link; }

    // Constant member function to retrieve the current data:
    value_type data( ) const { return data_field; }

    // Two slightly different member functions to retreive
    // the current link:

    const node* link( ) const { return link_field; }

    node* link( )             { return link_field; }

private:
    value_type data_field;
    node* link_field;
};

// FUNCTIONS for the linked list toolkit

std::size_t list_length(const node* head_ptr);

void list_head_insert(node*& head_ptr, const node::value_type& entry);

void list_insert(node* previous_ptr, const node::value_type& entry);

node* list_search(node* head_ptr, const node::value_type& target);

const node* list_search
    (const node* head_ptr, const node::value_type& target);

node* list_locate(node* head_ptr, std::size_t position);

const node* list_locate(const node* head_ptr, std::size_t position);

void list_head_remove(node*& head_ptr);

void list_remove(node* previous_ptr);

void list_clear(node*& head_ptr);

void list_copy(const node* source_ptr, node*& head_ptr, node*& tail_ptr);
```

```
// FILE: node1.cxx
// IMPLEMENTS: The functions of the node class and the
// linked list toolkit (see node1.h for documentation).
// INVARIANT for the node class:
//   The data of a node is stored in data_field, and the link in link_field.

#include "node1.h"
#include <cassert>    // Provides assert
#include <cstdlib>    // Provides NULL and size_t
using namespace std;

namespace main_savitch_5
{

    size_t list_length(const node* head_ptr)
    // Library facilities used: cstdlib
    {
        const node *cursor;
        size_t answer;

        answer = 0;
        for (cursor = head_ptr; cursor != NULL; cursor = cursor->link( ))
            ++answer;

        return answer;
    }


    void list_head_insert(node*& head_ptr, const node::value_type& entry)
    {
        head_ptr = new node(entry, head_ptr);
    }


    node* list_search(node* head_ptr, const node::value_type& target)
    // Library facilities used: cstdlib
    {
        node *cursor;

        for (cursor = head_ptr; cursor != NULL; cursor = cursor->link( ))
            if (target == cursor->data( ))
                return cursor;
        return NULL;
    }


    const node* list_search(const node* head_ptr, const node::value_type& target)
    // Library facilities used: cstdlib
    {
        const node *cursor;

        for (cursor = head_ptr; cursor != NULL; cursor = cursor->link( ))
            if (target == cursor->data( ))
                return cursor;
        return NULL;
    }

    }

.......

}
```