Week 7 Exercise Question 1 Solution

```
#include <iostream>
using namespace std;

// Linked List with Node struct

struct Node
{
    int data;
    Node *link;
};
typedef Node* NodePtr;


void head_insert(NodePtr& head, int the_number)
// Precondition: The pointer variable head points to the head of a linked list.
// Postcondition: A new node containing the_number has been added at the head of the linked list.
{
    NodePtr temp_ptr = new Node;
    temp_ptr->data = the_number;
    temp_ptr->link = head;

    head = temp_ptr;
}


NodePtr search (NodePtr head, int target)
// Precondition: The pointer head points to the head of a linked list. The pointer variable in the
// last node is NULL. If the list is empty, then head is NULL.
// Returns a pointer that points to the first node that contains the target. If no node contains the
// target, the function returns NULL.
{
    NodePtr here = head;

    if (here == NULL)
        return NULL;
    else
    {
        while (here->data != target && here->link != NULL)
            here = here->link;

        if (here->data == target)
            return here;
        else
            return NULL;
    }
}


void insert (NodePtr after_me, int the_number)
// Precondition: after_me points to a node in a linked list.
// Postcondition: A new node containing the_number has been added after the node pointed to by
// after_me.
{
    NodePtr temp_ptr = new Node;
    temp_ptr->data = the_number;
    temp_ptr->link = after_me->link;
    after_me->link = temp_ptr;
}
```

```
void remove (NodePtr& head, int num)  // remove all the nodes that contain the value num
// Precondition: The pointer variable head points to the head of a linked list.
{
    NodePtr before = head;
    NodePtr discard = head;
    while (discard != NULL)
    {
        if (discard->data != num)
        {
            before = discard;
            discard = discard->link;
        }
        else if (discard == head)
        {
            head = head->link;
            before = head;
            delete discard;
            discard = head;
        }
        else
        {
            before->link = discard->link;
            delete discard;
            discard = before->link;
        }
    }
}

void print_list (NodePtr head)
{
    for (NodePtr iter = head; iter != NULL; iter = iter->link)
        cout << iter->data << " ";
    cout << endl;
}

int main() // testing code
{
    NodePtr head = NULL;
    for (int i = 5; i > 0; --i)
    {
        // after you made your code work, switch the order of these two statements and test again.
        head_insert (head, 9);
        head_insert (head, i);
    }
    cout << "1 print: ";
    print_list (head);

    cout << "\n2 print: ";
    cout << search (head, 3) << endl;

    insert (head, 20);
    cout << "\n3 print: ";
    print_list (head);

    remove (head, 9);
    cout << "\n4 print: ";
    print_list (head);

    remove (head, 20);
    cout << "\n5 print: ";
    print_list (head);

    return 0;
}
```