Lab 2 - lab.h

```
#ifndef LAB_H
# define LAB_H

# include <iostream>
# include <fstream>
# include <catch.hpp>

using namespace std;

struct Entry
{
    Entry(string w="", string t="") : word(w), translation(t) {}
    bool    operator==(const Entry &e) const{
        return word == e.word && translation == e.translation;
    }
    string word;
    string translation;
};

struct Node
{
    Entry    entry;
    Node    *next;
};

bool    insert(Node *&head, Entry entry);
bool    loadDictionary(Node *&head, string fileName);
bool    add(string f, Entry entry);
bool    search(Node *&head, string word, Entry &e);
void    destroyList(Node *&head);

#endif
```

Libraries:
iostream: for for stdin/stdout via terminal
fstream: for read/write file

Structs:
Entry: for the array of words
Node: for the linked list

```cpp
#include <lab.h>

#ifndef UNIT_TEST

int main(void)
{
    Node *head = 0x00;
    string    file;

    cout << "Please, enter dictionary file name: ";
    getline(cin, file);
    if (loadDictionary(head, file))
    {
        string    input;
        string    w, t;
        Entry    e;

        while (!cin.eof())
        {
            bool    err = true;

            cout << endl;
            cout << "What do you want to do?" << endl
                      << "[1] Search a word" << endl
                      << "[ctrl + D] Exit" << endl;
            getline(cin, input);
            while (input == "1" && !cin.eof())
            {
                cout << endl;
                cout << "[ctrl + D] Exit" << endl
                     << "Please, enter a word: ";
                getline(cin, w);
                if (search(head, w, e))
                {
                    cout << endl;
                    cout << "English: " << e.word << endl
                         << "Italian: " << e.translation << endl;
                    err = false;
                }
                else if (!cin.eof())
                {
                    cout << endl;
```

Main:
In the main, there will be only communication part.
Other works will be handled in separte functions.

First, the program will ask for the dictionary file name and then store the data into linked list (loadDictionary fun.)
After that, it will ask for which words that users want to search.

If the word doesn't exist in the data,
it will ask for creating a new word.
If users create the word,
then it will be stored into the file.
It repeats the operations in a while loop until users hit ctrl+D.

```
                    cout << "Cannot find the word." << endl
                         << "[1] Look up another word" << endl
                         << "[2] Make a new word" << endl;
                    getline(cin, input);
                    if (input == "2")
                    {
                        t = "";
                        while (t.empty() && !cin.eof())
                        {
                            cout << endl;
                            cout << "Please, enter the translation for "
                                 << w << ": ";
                            getline(cin, t);
                            if (t.empty())
                            {
                                cout << endl;
                                cout << "Wrong input. Please try again."
                                     << endl;
                            }
                        }
                        destroyList(head);
                        if (!add(file, Entry(w, t)) || !loadDictionary(head, file))
                            cout << "Failed to add a new word." << endl;
                        err = false;
                    }
                }
            }
            if (err && !cin.eof())
                cout << "Sorry, wrong input. Please try again." << endl;
        }
    }
    else
        cout << "Failed to open file " << file << "." << endl;
    cout << endl;
    destroyList(head);
    return 0;
}

#endif
```

Lab 2 - insert.cpp

```cpp
#include <lab.h>

bool    insert(Node *&head, Entry entry)
{
    Node    *newnode = new Node;

    if (!newnode)
        return false;
    newnode->entry.word = entry.word;
    newnode->entry.translation = entry.translation;
    newnode->next = head;
    head = newnode;
    return true;
}

#ifdef UNIT_TEST

TEST_CASE("Testing list insert")
{
    Node    *head = 0x00;

    REQUIRE(insert(head, Entry("I", "io")));
    REQUIRE(head->entry.word == "I");
    REQUIRE(head->entry.translation == "io");

    REQUIRE(insert(head, Entry("a", "un")));
    REQUIRE(head->entry.word == "a");
    REQUIRE(head->entry.translation == "un");
}

#endif
```

insert:
This function takes the head of a list and
a content for a new node.
It creates a new node with the content and place it
at the front of the list (front of the head).
it returns false when it fails to allocate memory

Lab 2 - loadDictionary.cpp

```cpp
#include <lab.h>

bool    loadDictionary(Node *&head, string fileName)
{
    ifstream    ifs(fileName);
    string        title, w, t;
    bool        r = true;

    r = ifs;
    getline(ifs, title);
    while (ifs >> w >> t)
        if (!insert(head, Entry(w, t)))
        {
            r = false;
            break ;
        }
    return r;
}

#ifdef UNIT_TEST

TEST_CASE("Testing load list")
{
    Node    *head = 0x00;

    REQUIRE(loadDictionary(head, "Dictionary"));
    REQUIRE_FALSE(loadDictionary(head, "dictionary"));
}

#endif
```

loadDictionary:
This function takes the head of a list and a file name.
It will open, read the file, and store the data into the list.
It returns false when it fails to open the file

Lab 2 - search.cpp

```cpp
#include <lab.h>

bool    search(Node *&head, string word, Entry &e)
{
    for (Node *iter = head; iter; iter = iter->next)
    {
        if (iter->entry.word == word)
        {
            e = iter->entry;
            return true;
        }
    }
    return false;
}

#ifdef UNIT_TEST

TEST_CASE("testing search list")
{
    Node    *head = 0x00;
    Entry    e;

    REQUIRE_FALSE(search(head, "I", e));
    REQUIRE(loadDictionary(head, "Dictionary"));
    search(head, "I", e);
    REQUIRE(e.word == "I");
    REQUIRE(e.translation == "io");
}

#endif
```

search:
This function takes the head of a list and
an Entry struct that contains what users want to find.
It will iterate through the list and returns
true and store the matching translation into the Entry.
It returns false when it fails to find.

```cpp
#include <lab.h>

bool    add(std::string f, Entry entry)
{
    std::ofstream    ofs(f, std::ofstream::app);
    bool            r = ofs;

    if (r)
    {
        ofs << entry.word << " " << entry.translation << std::endl;
    }
    return r;
}

#ifdef UNIT_TEST

TEST_CASE("testing add new word")
{
    Node *head = 0x00;

    REQUIRE(loadDictionary(head, "Dictionary"));
    REQUIRE(add("Dictionary", Entry("word", "trans")));
}

#endif
```

add:
This function takes a file name and an Entry.
It will open the file and append the word and
the translation in the Entry into the file.
It returns false when it fails to open the file.

Lab 2 - destroyList.cpp

```cpp
#include <lab.h>

void destroyList(Node *&head)
{
    Node *t = NULL;

    while (head)
    {
        t = head;
        head = head->next;
        delete t;
    }
    head = NULL;
}

#ifdef UNIT_TEST

TEST_CASE("Testing list destroy")
{
    Node *head = NULL;

    REQUIRE(loadDictionary(head, "Dictionary"));
    destroyList(head);
    REQUIRE(head == NULL);
}

#endif
```

destroyList:
This function takes the head of a list.
It will delete all the allocated nodes while
iterating through the list.

Before

```
De wey% cat ../Dictionary
English Italian Dictionary
I io
a un
want volere
this questo
```

After

```
De wey% cat ../Dictionary
English Italian Dictionary
I io
a un
want volere
this questo
jkjl new
```

Result

```
De wey% valgrind ../lab02
==4264== Memcheck, a memory error detector
==4264== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==4264== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==4264== Command: ../lab02
==4264==
Please, enter dictionary file name: ../Dictionary

What do you want to do?
[1] Search a word
[ctrl + D] Exit
1

[ctrl + D] Exit
Please, enter a word: I

English: I
Italian: io

[ctrl + D] Exit
Please, enter a word: jkjl

Cannot find the word.
[1] Look up another word
[2] Make a new word
2

Please, enter the translation for jkjl: new

What do you want to do?
[1] Search a word
[ctrl + D] Exit
1

[ctrl + D] Exit
Please, enter a word: jkjl

English: jkjl
Italian: new

[ctrl + D] Exit
Please, enter a word:
==4264==
==4264== HEAP SUMMARY:
==4264==     in use at exit: 0 bytes in 0 blocks
==4264==   total heap usage: 61 allocs, 61 frees, 26,495 bytes allocated
==4264==
==4264== All heap blocks were freed -- no leaks are possible
==4264==
==4264== For counts of detected and suppressed errors, rerun with: -v
==4264== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```