# ICS 46
# Homework Assignment 2

**Programming Assignment**:

1. **(40 points – 20 points each)** Define a pair of related classes which implement unordered lists of std::string objects. Unordered means we don't care what order the entries are in, leaving the implementer freedom to choose where to insert and remove to allow efficient operations.

   Write two classes: UnorderedArrayList and UnorderedLinkedList.

   I. UnorderedArrayList must have three data members **buf**, **capacity**, and **size.**

   II. UnorderedLinkedList must have one data member **head**.

   DO NOT ADD ANY OTHER DATA MEMBERS.

   Methods will be **void insert(string word)**, **bool find(string word)**, **void remove(string word)**. You should also write **isEmpty** and **isFull** as well as the **constructor** and **destructor**. Note, you must use short circuited sequential search for both class find() methods.

   DO NOT ADD ANY OTHER METHODS TO YOUR CLASS IMPLEMENTATIONS.

   Also, allow duplicate entries in your unordered lists, but our tests will not test this because each word is in the input file only once.

2. **(40 points – 20 each)** Test both of your unordered list implementations by inserting all 45,392 words from the file random.txt (at the link below) then looking up all words from random.txt (they will all be there because you just inserted them all).

   Pass 45,500 as the initial size of your UnorderedArrayList constructor so it is plenty large to handle all these words without need for growing.

   Instrument your program using the provided Timer class in Timer.h (at the link below) so that it measures each of the following times:

   I. insert all words

   II. find all words

   III. remove all words

   For all of these, write plain functions, NOT methods. Do this for both of your implementations of unsorted list.

   Note, these will each make a pass over the input file to get the word to insert, find, or remove. DO NOT TAKE SHORTCUTS FOR remove_all_words - ever in this class. You must remove them one at a time by calling remove(word) for each of the 45,392 words.

   You can find random.txt (shuffled), words.txt (sorted), and a timer class Timer.h at this URL. Timer.h includes a sample of how to use it at the bottom in a comment.

3. **(20 points)** Write the time complexity, using big-oh notation, next to each method in UnorderedArrayList, UnorderedLinkedList, and your testing functions (e.g., insert_all_words, find_all_words, remove_all_words, main).

   **Minus 5 points for each one missing or incorrect, up to 20 points.**

**<u>How to Design Your Program</u>**:
Here is an outline of the structure you should use. It may be tempting to introduce a base class for UnorderedList so you can write polymorphic functions for most of the test functions but then the time complexities will be variable depending on the type of list passed in. Note, we will do this in a future homework assignment.

```
// includes and "using"

class UnorderedArrayList
{
	string * buf;
	int capacity;
	int size;
public:
	UnorderedArrayList(int max_len);
	void insert(string word);
	bool find(string word);
	void remove(string word);
	void print(ostream & out);
};

class UnorderedLinkedList
{
	struct ListNode {
		string info;
		ListNode * next;
		ListNode(string new_info, ListNode *new_next) :
			info(new_info), next(new_next) { }
		static void print(ostream & out, ListNode *L)
		{
			if (L)
			{
				out << L->info << endl;
				print(out, L->next);
			}
		}
	};
	ListNode * head;
public:
	UnorderedLinkedList();
	void insert(string word);
	bool find(string word);
	void remove(string word);
	void print(ostream & out);
};

ostream & operator << (ostream & out, UnorderedArrayList & L)
{
	L.print(out); return out;
}

ostream & operator << (ostream & out, UnorderedLinkedList & L)
{
     L.print(out); return out;
}

void insert_all_words(string file_name, UnorderedArrayList & L)
{
/*
     declare timer object
     open file
     start timer
     for each word, w, in file
          L.insert(w);
     stop timer
     close file
     report time
*/
}

void find_all_words(string file_name, UnorderedArrayList & L)
{
}

void remove_all_words(string file_name, UnorderedArrayList & L)
{
}

void insert_all_words(string file_name, UnorderedLinkedList & L)
{
}

void find_all_words(string file_name, UnorderedLinkedList & L)
{
}

void remove_all_words(string file_name, UnorderedLinkedList & L)
{
}

void test_UnorderedArrayList_methods(string file_name, UnorderedArrayList
& L)
{
     cout << “Testing UnorderedArrayList:” << endl;
     insert_all_words(file_name, L);
     find_all_words(file_name, L);
     remove_all_words(file_name, L);
}
```

```
void test_UnorderedLinkedList_methods(string file_name,
UnorderedLinkedList & L)
{
     cout << "Testing UnorderedLinkedList:" << endl;
     insert_all_words(file_name, L);
     find_all_words(file_name, L);
     remove_all_words(file_name, L);
}

int main(int argc, char * argv[])
{
     char *input_file = argc == 2 ? argv[1] : "random.txt";
     UnorderedArrayList UAL(45500);
     test_UnorderedArrayList_methods(input_file, UAL);
     UnorderedLinkedList ULL;
     test_UnorderedLinkedList_methods(input_file, ULL);
     return 0;
}
```

Output of your program will be three times for each UnorderedList: insert_all_words time, find_all_words time, and remove_all_words time. Because there are two implementations of UnorderedList (UnorderedArrayList and UnorderedLinkedList), <u>you will print 6 times</u>.

I suggest you create a smaller random.txt for initial testing. Once your program seems to be working correctly, you can run in on the full random.txt file.

- Submit two things:
  1. A single report in .pdf format that shows your program works correctly when run under valgrind. Your report must follow the specified format, then submitted to GradeScope.
  2. A zip of your program which includes Makefile that builds your program with the required warning flags.
- Copy/Paste your code for the functions (with their time complexity) related to insert_all_words() and find_all_words() for each of the two different implementations of UnorderedList, (should be about 6 functions), e.g.,
  - UnorderedArrayList::insert(string word); // O(?)
  - UnorderedLinkedList::insert(string word); // O(?)
  - insert_all_words(UnorderedArrayList & L); // O(?)
  - insert_all_words(UnorderedLinkedList & L); // O(?)
  - UnorderedArrayList::find(string word); // O(?)
  - UnorderedLinkedList::find(string word); // O(?)
  - find_all_words(UnorderedArrayList & L); // O(?)
  - find_all_words(UnorderedLinkedList & L); // O(?)

Homeworks that are not formatted nicely will not be accepted. Pay careful attention to memory management. Do not share memory across objects. Obvious code copying, from a textbook, from another student, or within your own program, will be downgraded (write everything once (perhaps by writing auxiliary functions), then call the function or method each time that operation is required).

Feel free to add additional functions or private class methods, but please do not add any more data members to the class and do not add any public methods.