# Assignment 1

Sep 6th

Zhihan Guo

## Bio

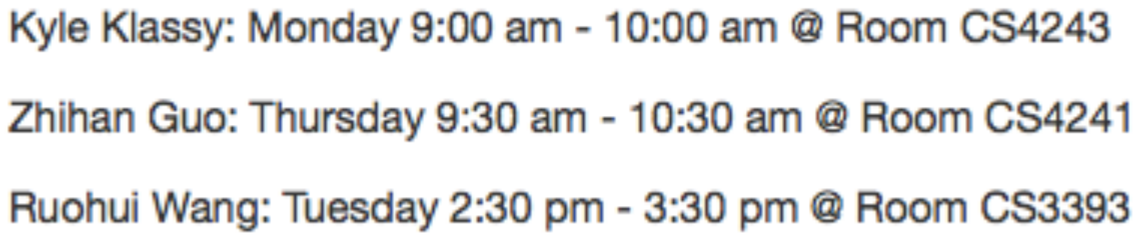
- Zhihan (Scarlet) Guo
- Zhi - “G”
- Email: zhihan “at” cs.wisc.edu
- Office: 4241

## Announcements

- Course Website: https://kyle-klassy.github.io/cs564-fall19/
  - TA Office Hours
  - Piazza
  - Lecture Notes:
    - **Dropbox**
  - Assignment 1 (Due Next Sunday, **Sep 15th @ 11:59PM**)
    - Individual
- Canvas
  - Still working on it..

- Friday slides will be posted on course website

## TA Office Hours



Kyle Klassy: Monday 9:00 am - 10:00 am @ Room CS4243

Zhihan Guo: Thursday 9:30 am - 10:30 am @ Room CS4241

Ruohui Wang: Tuesday 2:30 pm - 3:30 pm @ Room CS3393

## Piazza

- Announcements
- Assignment Clarifications
- Q & A

**DO NOT POST ANY CODE PUBLICLY:**

Students may **NOT** publicly post any code that is part of any assigned problem (working or otherwise). If you cannot ask your question without including code, you must mark your question as private and visible only to the course Instructors (which includes TAs). This will help us all avoid unnecessary academic misconduct concerns and consequences.

## Overview

- Assignment Description & Demo
- Developing Tools
    - Programming Tool
    - Development Platform
    - Running, Testing and Debugging
- Q & A

## Overview

- **Assignment Description & Demo**
- Code Development
  - Programming Tool
  - Development Platform
  - Running, Testing and Debugging
- Q & A

## Assignment 1: Word Locator in C++

Goal:

- help you brush up your C++ programming skills and refresh knowledge of data structure (CS367/CS300,400)

Description:

- develop a “word locator” program written in C++, which will allow a user to check if a specified (re)occurrence of a specified query word appears in the input text file.

# Word Locator

Given a text document, your program should be able to

- **“load”** the document. Scan the document ; parse and store the words in a data structure.

## Demo

1 2 3 4 5

Sing a song of sixpence,
A pocket full of rye;
Four and twenty blackbirds
Baked in a pie.

15 16 17 18

```
>load sixpence.txt
```

## Word Locator

Given a text document, your program should be able to

- **“load”** the document. Scan the document ; parse and store the words in a data structure.
- **“locate”** the $n_{th}$ occurence of a word. Given a word, return the  position of the nth occurrence of the word in the document you load.
- **“new”**. Reset the word list to original (empty) state.
- **“end”**. Terminate the program

## Demo

1 2 3 4 5

Sing a song of sixpence,
A pocket full of rye;
Four and twenty blackbirds
Baked in a pie.

15 16 17 18

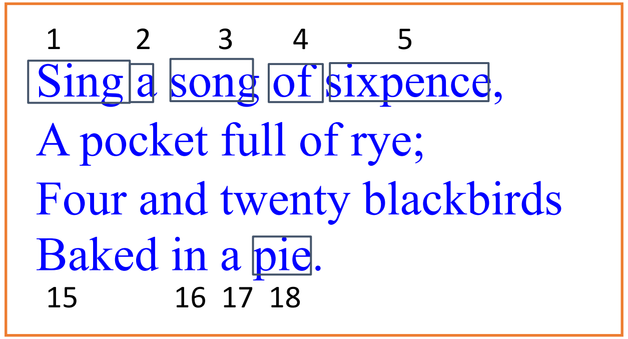
```
>load sixpence.txt
>locate song 1
3
>locate Song 1
3
>locate SoNg 1
3
>locate pie 1
18
```

## Word Locator

Given a text document, your program should be able to

- **“load”** the document. Scan the document ; parse and store the words in a data structure.
  - E.e. “load sample.txt”
- **“locate”** the $n_{th}$ occurence of a word. Given a word, return the position of the nth occurrence of the word in the document you load.
  - E.g. “locate word 1”

# Demo



Sing a song of sixpence,
A pocket full of rye;
Four and twenty blackbirds
Baked in a pie.

```
>load sixpence.txt
>locate song 1
3
>locate Song 1
3
>locate SoNg 1
3
>locate pie 1
18
```

## Word Locator

Given a text document, your program should be able to

- **“load”** the document. Scan the document ; parse and store the words in a data structure.
- **“locate”** the $n_{th}$ occurence of a word. Given a word, return the position of the nth occurrence of the word in the document you load.
- **“new”**. Reset the word list to original (empty) state.
- **“end”**. Terminate the program

## Demo

```
>load sixpence.txt
>locate song 1
3
>locate Song 1
3
>locate SoNg 1
3
>locate pie 1
18
>new
>locate song 1
No matching entry
>end
```

## Handle Incorrect Commands (check assignment page!)

- If a bad command is entered, print “ERROR: Invalid command”, and go to the next prompt.
- Examples of bad command
  - Invalid command. E.g. “find word 7”
  - Invalid words. E.g. “rats#”
  - Extraneous content. E.g. “locate word 5 7”
- Other notes:
  - if an incorrect load command is entered, such as “load” (no filename) then your data structure should not be reset.
  - Commands are case insensitive. “LoCaTe word 1” is a valid command.

## Choices of Data Structure

- You CAN use C++ Standard Template Library (STL)
    - a set of C++ template classes to provide common programming data structures and functions
- Use Unordered Associative Containers: unordered set, map, etc.
- Implement Tree-based Structure using containers provided by STL

## Overview

- Assignment Description & Demo
- **Code Development**
    - Programming Tool
    - Development Platform
    - Running, Testing and Debugging
- Q & A

# Programming Tool for C++

- Check course webpage for tutorials and IDEs.
    - Additional materials: http://pages.cs.wisc.edu/~gerald/cs368/
        - Lecture notes, resources, etc.

## Development Platform

- Ubuntu ~~14.04~~ 18.04 LTS Linux
- CSL Machine, need a cs account
- See Assignment Page

# Getting Started – download files

- The files for this assignment are located in http://pages.cs.wisc.edu/~jignesh/cs564/projects/wc/ This directory has the following files:
- **wl.h and wl.cpp**: Empty files in which you have to add your solution code.
- Makefile: A sample makefile.
- sixpence.txt: A sample text file.
- sixpence.cmd: A sample command file.
- sixpence.out: Sample output when the command “wl < sixpence.cmd” is run.
- wrnpc.txt: Another sample text file (sample command and outputs are not provided for this file).

## Running

- Step 1: compile using provided Makefile
  - “make all”
- Step 2: run executable and enter commands
  - “./wl”

## Testing

- Use provided sample commands to test and compare the output with sample output (sixpence.out):
    - ./wl < sixpence.cmd
- Use the larger sample document (wrnpc.txt) to design your own commands and check if the behavior is as expected.

- your assignment will be tested against our (more comprehensive) test driver.

- You are encouraged to develop additional tests on your own.

## Debugging

- As flag ‘-g’ is provided in Makefile, you can use gdb to debug your program.
    - “gdb wl”
    - Check basic command: [http://pages.cs.wisc.edu/~horwitz/gdb/gdb.ps](http://pages.cs.wisc.edu/~horwitz/gdb/gdb.ps)

## Documentation

- Your code should be fully commented following the specs for Doxygen (www.**doxygen**.org). In other words, you should be able to generate documentation for your code using doxygen.
- An example of the documentation generated using doxygen:
    - http://pages.cs.wisc.edu/~jignesh/cs564/projects/BadgerDB/BufMgr/docs/annotated.html

## Submission

- More details about the submission procedure will be posted next week.