Discussion Section 10

- HW 9

- Regular expressions

- sed

- AWK

- Version control (Git)
Homework 9 questions?
Homework 9 questions?

- In C++, use srand() and rand()
Homework 9 questions?

- In C++, use srand() and rand()
  - srand() sets the seed, rand() generates a random number using the seed
Homework 9 questions?

• In C++, use srand() and rand()
  – srand() sets the seed, rand() generates a random number using the seed
    • For testing, you can use srand() to set the same seed
Homework 9 questions?

- In C++, use srand() and rand()
  - srand() sets the seed, rand() generates a random number using the seed
    - For testing, you can use srand() to set the same seed
    - For actual running, set to something like the system clock time (i.e. time(NULL))
Homework 9 questions?

• In C++, use srand() and rand()
  – srand() sets the seed, rand() generates a random number using the seed
    • For testing, you can use srand() to set the same seed
    • For actual running, set to something like the system clock time (i.e. time(NULL))
    • rand() gives integers between 0 and RAND_MAX
Homework 9 questions?

- In C++, use `srand()` and `rand()`
  - `srand()` sets the seed, `rand()` generates a random number using the seed
    - For testing, you can use `srand()` to set the same seed
    - For actual running, set to something like the system clock time (i.e. `time(NULL)`)
    - `rand()` gives integers between 0 and `RAND_MAX`
      - to get a number between 0 and 1, just divide by `RAND_MAX`
Regular Expressions

- Strings that define a pattern, often used for searching and matching
Regular Expressions

- Strings that define a pattern, often used for searching and matching
- Many special operators to define patterns
Regular Expressions

- Strings that define a pattern, often used for searching and matching
- Many special operators to define patterns
  - '^', '$', '.', '*', '+', '?' to name a few
Regular Expressions

- Strings that define a pattern, often used for searching and matching
- Many special operators to define patterns
  - '^', '$', '.', '*', '+', '?' to name a few
- Examples:
Regular Expressions

- Strings that define a pattern, often used for searching and matching
- Many special operators to define patterns
  - `'^', '$', '.', '*', '+', '?'` to name a few
- Examples:
  - `'a'`: anything with the letter 'a'
Regular Expressions

• Strings that define a pattern, often used for searching and matching

• Many special operators to define patterns
  – '^', '$', '.', '*', '+', '?' to name a few

• Examples:
  – 'a': anything with the letter 'a'
  – 'a.b': anything with the letters 'a' and 'b' separated by any character
Regular Expressions

• Strings that define a pattern, often used for searching and matching

• Many special operators to define patterns
  – '^', '$', '.', '*', '+', '?' to name a few

• Examples:
  – 'a': anything with the letter 'a'
  – 'a.b': anything with the letters 'a' and 'b' separated by any character
  – '^a.*z$': anything that starts with the letter 'a' and ends with the letter 'z'
Regular Expressions

- Strings that define a pattern, often used for searching and matching
- Many special operators to define patterns
  - '^', '$', '.', '*', '+', '?' to name a few
- Examples:
  - 'a': anything with the letter 'a'
  - 'a.b': anything with the letters 'a' and 'b' separated by any character
  - '^a.*z$': anything that starts with the letter 'a' and ends with the letter 'z'
  - '\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b': email addresses
sed (**stream editor**)  

- Changes text (in file or piped directly to sed)
sed (stream editor)

- Changes text (in file or piped directly to sed)

- Most often used for substitution with the 's' command (though there are other commands)
sed (stream editor)

- Changes text (in file or piped directly to sed)

- Most often used for substitution with the 's' command (though there are other commands)
  - sed 's/old/new/' filename
sed (stream editor)

- Changes text (in file or piped directly to sed)

- Most often used for substitution with the 's' command (though there are other commands)
  - sed 's/old/new/' filename

- If you want to know more, here's a really good in-depth tutorial:
AWK

- Useful for handling tables of data in text format (csv, tsv, etc.)
AWK

- Useful for handling tables of data in text format (csv, tsv, etc.)
- Can both extract data and process it
AWK

- Useful for handling tables of data in text format (csv, tsv, etc.)
- Can both extract data and process it
- Basic structure of an AWK command:
  - BEGIN {stuff before you look at file}
  {stuff while looking at file}
  END {stuff after you're done with the file}
AWK

- Useful for handling tables of data in text format (csv, tsv, etc.)
- Can both extract data and process it
- Basic structure of an AWK command:
  - BEGIN {stuff before you look at file}
    {stuff while looking at file}
    END {stuff after you're done with the file}
- Same person has another good tutorial:
Version Control (Git)

- Version control is useful for a variety of reasons that all boil down to keeping track of code and changes to that code.

- You can use Git (and other version control systems) both on your own and in collaborative projects.
Solo Git
Keeping track of previous code

Old Code
Keeping track of previous code

Keeping track of previous code

Old Code → New Code/Bug Fixing → New Code

Reasons to revert:
Keeping track of previous code

New Code/Bug Fixing

Old Code → New Code

Reasons to revert:

- New bug, better to restart from old code
Keeping track of previous code

Reasons to revert:

- New bug, better to restart from old code
- Need to regenerate results with previous version of code
Keeping track of previous code

Old Code → New Code / Bug Fixing → New Code

Reasons to revert:

- New bug, better to restart from old code
- Need to regenerate results with previous version of code
Developing/maintaining public software

v1.0
Developing/maintaining public software

v1.0

New Feature
Developing/maintaining public software

v1.0 -> Hotfix

Hotfix -> New Feature
Developing/maintaining public software

- **v1.0**
- **Hotfix**
- **v2.0**
- **New Feature**
Collaborative Git
Branches allow testing and parallel collaboration
Group software development

v1.0 → Hotfix

New Feature

v2.0

New Feature
New project/software, same starting point or shared resources
New project/software, same starting point or shared resources

Maybe I can modify/use this code for something else
New project/software, same starting point or shared resources

Maybe I can modify/use this code for something else
New project/software, same starting point or shared resources

Maybe I can modify/use this code for something else

Project A ➔ Project B ➔ Updated Project A ➔ Up-to-date Project B
New project/software, same starting point or shared resources

Maybe I can modify/use this code for something else
Solo Commands

• Most important for solo work:
  – Creating a repo (git init)
  – Adding, deleting (git add, git rm)
  – Committing (git commit [-m])
  – Undoing changes (git reset [file])
  – Checking what files you've changed (git status)
  – Looking at the change log (git log [--stat] [-p])
  – Ignoring files (.gitignore)
Collaboration Commands

• Most important for collaborations:
  – Cloning a repo (git clone [address])
  – Pushing/Pulling (git push/pull [branch])
  – Safely undoing changes for everyone (git revert [commit])
  – Making a new branch (git branch [name])
  – Switching branches (git checkout [name])
  – Merging branches (git merge [name])