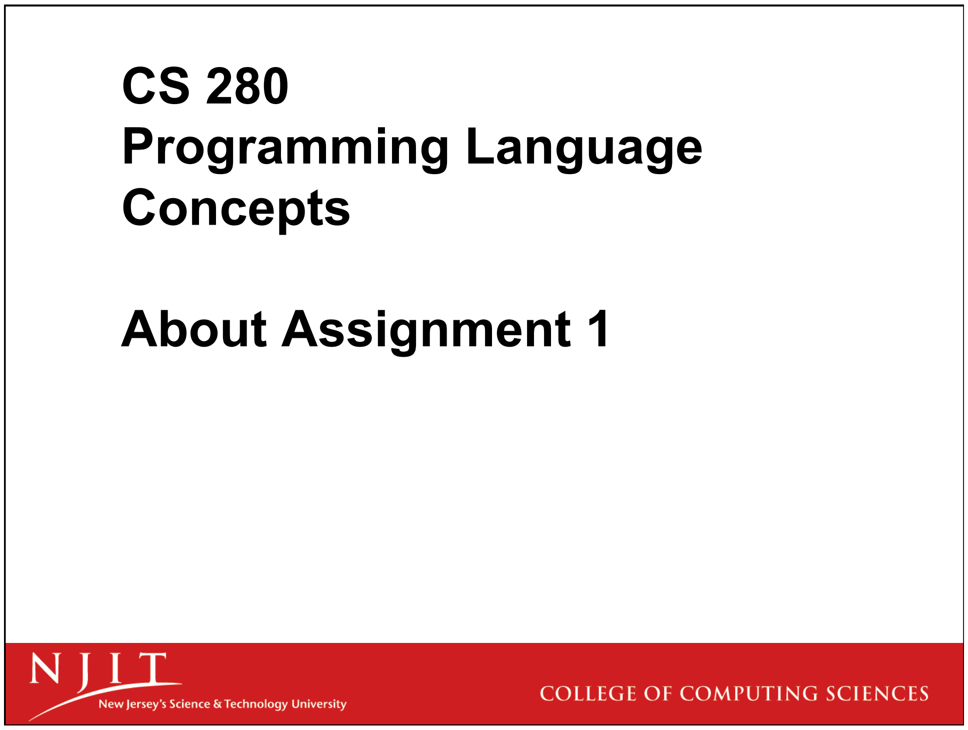
NJIT
New Jersey's Science & Technology University
THE EDGE IN KNOWLEDGE



CS 280 Programming Language Concepts

About Assignment 1

NJIT New Jersey's Science & Technology University
COLLEGE OF COMPUTING SCIENCES

Notes for Assignment 1

- Be sure to understand the assignment
- Decide on the information that you need to keep track of to do the assignment
  - "data structures programs"
- What algorithm should you use?
- How should you save the data

Understand the assignment

- The command line flags specified in your assignment change the behavior of the program.
  - You have to check for the presence of the flags, remember that they were specified, and change the behavior of the program if they are specified
- You're finding the "longest line" and the "longest word"
  - One complicating factor on the "longest line" question is that the definition of length of a line is different if the user specifies the −b flag
- Note that our idea of the "longest line" and "longest word" will probably change during the course of processing the input
  - this is similar to the classic "find the largest integer in the list" problem

Information To Keep Track Of

- What flags were specified?
- What is our current idea of the length of the longest line?
- What is our current idea of the length of the longest word?
- What are the longest words?

Information To Keep Track Of

- What flags were specified?
  - you can use a boolean variable for each possible flag argument
- What is our current idea of the length of the longest line?
  - Remember the length of the longest line in an integer
  - Note that there might be two ways to calculate the length of the line
- What is our current idea of the length of the longest word?
  - Remember the length of the longest word in an integer

What are the longest words?

- We need some sort of a container where we can store the longest words
  - We also want to keep track of how many times we see each of the longest words
- Choices:
  - vector
    - PRO: easy to add things and iterate through words, easy to clear out the list
    - CON: to keep track of word counts, would need to iterate through the vector and keep a count for each unique word somewhere
  - map
    - PRO: easy to add, iterate, and clear
    - CON: slightly more expensive to add things to the map, but all the necessary counting and calculation and management of duplicate words is done for you

Algorithm(s)

- Finding the longest line is like finding the largest integer in a list:
  - keep track of your program's idea of the length of the longest line in some integer variable
  - initialize the integer to 0
  - every time you have a line, figure out its length. If the length is longer than your current idea of the length of the longest line... then this line's length is the new value of the length of the longest line
  - when the program ends, the length of the longest line is in the integer variable you created
  - It's easy to prove this is correct

Algorithm(s)

- Keeping track of the number of lines that have length == the longest line is straightforward
  - initialize a counter to 0
  - when you see a new value for "length of longest line", set the counter to 1
  - when you see the same value for "length of longest line", add 1 to the counter
  - At the end of the program, the counter contains the number of times you saw a line whose length is that of the longest line in the file

Algorithm(s)

- Finding the length of the longest word uses the same algorithm as the length of the longest line
- However you also must remember the longest words, and have a count for each of them
- Alternatives:
  - just go through and count how many occurrences there are for every word in the file
    - at the end, scan through the map to find the largest length, then scan through again to print the words with that largest length
  - keep a map that counts the number of times each of the "longest words" appears. If during the run of your program, you change what you think the length of the longest word is, throw away the old contents of your map
    - more work during the run of the algorithm, but less work at the end
    - probably could show that this is less computationally complex in terms of space and time

Algorithm(s) for flags

- -c is easy
  - just print the counts if the flag was specified
- -m is pretty easy
  - Before printing out results, you have to scan through the “longest words” and find the maximum frequency, then only print out the longest words that have the maximum frequency
- -b is the most complex
  - without −b, the length is just the length!
  - with −b, you need to consider multiple consecutive spaces as a single space when you calculate the length
  - two possible solutions
    - write code to walk through the string containing the line. count characters, but just count consecutive whitespace one time
    - when breaking up the line into words, you can say that the length of the line for the purposes of −b is a function of the sum of the length of each of the words in the line PLUS the number of words in the line MINUS 1
      - there has to be at least one space between each of the words in a line, by definition
      - for −b you want to view it as if there is only one space between words
      - two words have one space between them. Three words have two spaces, one between word 1 and word 2, and one between word 2 and word 3, etc

NJIT New Jersey's Science & Technology University COLLEGE OF COMPUTING SCIENCES

NJIT THE EDGE IN KNOWLEDGE