CSC 207 2014S: Extra Session, Week 5

Overview

- Admin
- You ask questions.
- I try to give answers.

Admin

- Welcome to the people who live on the 3rd floor, or at least seem to.

Questions

Why does my code give different output when given the same inputs?

Sentient and malicious. I'll look at it individually when I can find time.

Why do you write randomized tests?

They are fun to write.

They tend to catch things that I wouldn't otherwise think about.

It's a bit easier to write some really tricky things.

But, as you've noted, they don't necessarily give much insight other than "your code sux".

They are also hard to replicate.

Formatting and commenting conventions.

- Most things are just conventions; they do not affect readability (except when violated).
  - Assumption: identation shows nesting
- What Sam saw yesterday.

  if (test) { /* whatever */ *doStuf();* */* whatever */ }
- In an ideal world, you think about what you want to do and have an English language-model. Comments show that

  // Gather together all the information // If I have a nontrival amount of information // Throw away the irrelevant parts

- Next, you write the code the implements the goals

  // Gather together all the information this.stuff = sendOutRequestToTheServer("..."); this.stuff.append(localInformation); // If I have a nontrival amount of information if (this.stuff.length > REALLY_BIG) { // Throw away the irrelevant parts this.stuff = Arrays.subArray(this.stuff, 0, 10); }

- In addition to violating the model
  - Can be hard to read; it's hard to maintain consistency
  - Leads to long lines
  - Traditionally leads to subtle bugs

    if (test) /* whatever doStuf(); /* whatever */ doMoreStuf();

- When are side comments okay?
  - Declare variables

Can you help us understand the MutableString class a bit better?

We're storing all of the characters in an array. Assumption: We have more control over moving things around; might allow us to be more efficient.

Two things to worry about: subarray that represents the string; extra space

A sophisticated implementation might store index of first character of interest and index of last character of interest.

We'll assume that first `size` characters are the stuff of interest; everything else is for when we expand.

```
/**
 * The characters in the string plus some optional extra space in case
 * we expand the string.
 */
char contents[];

/**
 * The number of characters in the string.  We assume they are stored
 * in positions 0 .. this.size-1
 */
int size;
```

Append adds to the end of the string

```
// Note how many characters are in the appendage
int applen = appendage.length();
// Figure out how much space the new string will take
int newsize = this.size + applen;
// If there's not enough room in the array
if (this.contents.length < newsize)
  {
```

```
    // Build the new array
    character newcontents[] = new character[computeNextCapacity(newsize)];
    // Copy over characters
    for (int i = 0; i < this.size; i++)
      {
        newcontents[i] = this.contents[i];
      } // for
    // And use the new array
    this.contents = newcontents;
// Copy the characters from the appendage (note: Sam is bad at getting
// the math right)
for (int i = 0; i < applen; i++)
  {
    this.contents[this.size + i] = appendage.charAt(i);
  } // for
// Update the size
this.size = newsize;
```

You said something about that array copy leaking memory in C. Can you explain?

> char contents[] = malloc(16 * sizeof(char)); ... char newcontents[] = malloc(A*BIG*NUMBER); ... contents = newcontents;

How do you delete?

```
// Identify where we are copying from and where we are copying to
int target = ...;
int source = ...;
// While there are characters left to copy
while (source < this.size)
  {
    // Copy a character
    this.contents[target] = this.contents[source];
    // And move on
    ++target;
    ++source;
  } // while
```

Copyright (c) 2013-14 Samuel A. Rebelsky.

This work is licensed under a Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit `http://creativecommons.org/licenses/by/3.0/` or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.