CSCI 2270 Data Structures and Algorithms 2/26/2014 BigNums

Elizabeth White
elizabeth.white@colorado.edu
Office hours: ECCS 128 or ECCS 112
Wed 9:30am-11:30am
Thurs 10:00am-11:00am

Administrivia

- First midterm exam is in grading
- Read Recursion (chapter 4); concentrate on factorial and binary search sections for now
- This week’s lab is on LCS alignment
- If you are BORED and you want to go for extra credit, figure out how to extend this to backtrack and display the matching for the 2 input strings, and show it to me next week.

Seminar attendance

- For your writeups of the seminars, I am happy if you track the talk, but I want a little more reflection from you along with a summary. (No offense, but straight summaries are not fun to read, and this is 5% of the grade, so I can be a little more demanding.)
- I'm looking for 6 double spaced pages, by email, plus a couple of references to journals or magazines, and at least half of this paper should talk about your reaction to the work. For instance, is there a new direction you can take this idea? Or did you have to do extra work to learn more about something mentioned in the talk, and did you learn anything new? *Tell me something interesting that you learned*.

BigNum numbers

- Member variables:
  - unsigned int capacity;   // how big your digits array is
  - unsigned int used;   // how much of your array is not garbage
  - unsigned int* digits;   // array of unsigned ints (0-9)
  - bool positive;   // true for numbers >= 0

- We store these digits backwards, which is really handy later. Don't store them in forwards order (it's too much work for us to debug).
- Watch the odd behavior of unsigned ints (and don't even think of changing these to regular ints; that is not allowed).

What happens in the code below?

```
BigNum a;                             // default constructor
BigNum b = 786;                       // int constructor
BigNum c = b;                         // copy constructor
BigNum d = (string) "9287878";  // string constructor
c = a;                                // assignment operator
b = b;                                // assignment operator
```

int constructor

Is the input int positive or negative? Set the BigNum's positive.

    If the input int's negative, make it positive

How many decimal places does the number have?

    How many times can you divide it by 10 before it disappears? That's your used.

    How do you slice off the digits? Modulus.

    Should you rely on DEFAULT_CAPACITY? No. Count.

Will this work for a zero int? No; handle that separately.

    Note: assignment is handy here.

string constructor

How long is the string? Use strin.length() to make a guess at used.

What if you have a - or + sign? Count used down by 1.

How do you get the digits out?

```
strin[i] is '2'.                // the letter 2
strin[i] - '0x30' is 2.         // the number 2
strin[i] - '0' is 2.            // the number 2
strin[i] - '48' is 2.           // the number 2
```

The trick here is reading the string in the forward direction and placing the digits in the backward direction in the right slots.

Work this out on paper first.