CSCI 2270
Data Structures and Algorithms
Lecture 10

Elizabeth White
elizabeth.white@colorado.edu
Office hours: ECCS 128
Wed 1-2pm
Thurs 2-3pm

Admin

HW1 due on Monday, 11:55 pm: Singly Linked List
This will be *interview graded.*

Good colloquium talk this week, 3:30-4:30 Thursday, ECCR 265

Exam review:
	slides and quiz questions, labs, homework
	insertion sort example

Programming exam: in lab next Tuesday/Thursday.
	2 problems will be posed; you will answer one
Written exam: in class next Friday

Factorial of n

How to compute factorial of a positive integer n? A loop

:

```
unsigned int factorial_1(unsigned int n)
{
        unsigned int product = 1;
        unsigned int counter = 1;
        while (counter < n)
                product *= counter++;
        return product;
}
```

Factorial of n

How to compute factorial of a positive integer n? Recursion

```
unsigned int factorial_2(unsigned int n)
{
        if (n < 2)
                return 1;
        else
                return n * factorial_2(n - 1);
}
```

Factorial of n

Recursive functions call themselves….

```
unsigned int factorial_2(unsigned int n)
{
	if (n < 2)			// base case
		return 1;
	else				// more recursion possible
		return n * factorial_2(n - 1);
}
```

Problem HAS TO GET SMALLER in each recursive call

Not clearing a linked list

Another way… THIS SLIDE WAS WRONG (thanks to the 2 students who argued with me after class!)

```
void clear_list(node*& head_ptr)
{
        if (head_ptr == nullptr) return;
        if (head_ptr->next == nullptr)
        {
                delete head_ptr; head_ptr = nullptr; return;
        }
        clear_list(head_ptr->next);
}
```

Actually clearing a linked list

Another way…

```
void clear_list(node*& head_ptr)
{
        if (head_ptr == nullptr) return;
        clear_list(head_ptr->next);
        delete head_ptr;
        head_ptr = nullptr;
        return;
}

// check by printing head_ptr->data before deleting
```

Recursion

Recursive factorial is not so high performance…

- Recursive calls cost time
- Recursive functions with local variables cost *lots of space*

But other recursive methods are very useful. Examples:

- Binary search of sorted array
- Solution to Towers of Hanoi problem
- Quicksort/heapsort/mergesort to sort arrays
- Processing fractals: Koch snowflake, Sierpinski triangle…

Practice questions for exam…

Write me a function that takes 2 unsigned ints, m and n, and creates a 2-dimensional array of integers on the heap with m rows and n columns.

Write me a function that destroys the above 2-dimensional array.

Write me a function that tells me if 2 sorted int_arrays have exactly the same numbers.

Write me a function that tells me if 2 unsorted int_arrays have exactly the same numbers. Assume you can’t sort the arrays. (Why is this so much harder?)

Practice…

Write me a function to convert a decimal number to a string of binary digits. This requires working out how many digits you will need, and if you use the algorithm on my slides, you'll need to write the digits into the string backwards. Return the string.

Write me a function to convert a string representing a binary number to a decimal number. Return this number as an int.

Write me a function void minimum(bool& empty, int& answer) to find the smallest number in an unsorted int_array's data if the array's not empty, and return 0 if it is.

Practice…

Write me a function that takes an int_array as input, and then makes a new array 10 times larger, with the contents of the original data array repeated 10 times. Replace the original int_array's data with this new array, but don't leak memory.

Write me a recursive function to copy a singly linked list (from the end node backwards.)

Write me a recursive function to count the number of times a target integer appears in a singly linked list.

Practice…

Write me a recursive function to take a singly linked list and return a new list with the same elements as the old one, but in reverse order.