CSI 310: Lecture 15

2nd half of this course will cover:

- Trees, Recursion, Stacks, and
  (context-free language) Expressions
  as 4 views of the SAME IDEA

- Maze (graph) search/traversal

Reading: DSO ch. 7, 8, 9 and 10.
First, note that a local variable.

For example, running fact(7) computes \( 7 \times 6 = 42 \) and returns \( 42 \), which returns

\[
\{
\text{if } (n == 0) \text{ return } 1;
\text{else return } \text{fact}(n-1)*n;
\}
\]

int fact(int n)

Example: When the function runs, calls the same function, either directly or indirectly.

Definition: A function is recursive means the body of the function sometimes,

re-implement.

It is worth it.

3) Elegant way to write programs; performance can be improved with routine.

2) Powerful problem solving technique.

1) Understanding, not just programming, data structures, and algorithms.

Recursion: trees, recursion, expressions, stacks are closely related.

University at Albany Computer Science Dept.
The store used for AUTOMATIC VARIABLES is in the activation record.

The EVENT of "calling one function once" is an ACTIVATION.

Each time a function is CALLED, an ACTIVATION RECORD is created.
RETURNED to.

4. Now, automatic variables are in the activation record of the activation.

RETURNED to.

3. The activation whose CALL originally created this activation is the activation "goes away".

2. This activation's activation record.

1. The return value (if any) is saved for use by the caller.

When a function activation executes the return statement:
Only true "logically"... this data is actually stored in the called activation.

**ACTIVATION** is destroyed, and its Record gets recycled.

**Really:** When an **ACTIVATION** executes the **return** operation, that

```
        (3) The return
```

**Definition of Activation Record:** The data structure that holds (1) An

**Really:** A new **Function Activation** is created, and the function's body

**Wrong:** Control "jumps" or "goes to" the function's body.

**What Happens when the computer executes a FUNCTION CALL?**

**Really:** When an **ACTIVATION** executes the **return** operation, that

```
        (3) The return
```

**Definition of Activation Record:** The data structure that holds (1) An

**Really:** A new **Function Activation** is created, and the function's body

**Wrong:** Control "jumps" or "goes to" the function's body.
\{
    return Solution;
\}

Solution = MERGE( ANST1, ANST2 );
\}
dnode * ANST = MERGE( Superproblem2 );
dnode * ANST = MERGE( Superproblem1 );
\} // RECURSE one or more times:
Superproblem2 = INSTANCE; // SPLIT removed half the original list.
Superproblem1 = SPLIT( INSTANCE );
dnode * Superproblem1, Superproblem2;
\} else
\{
    Solution = INSTANCE; // TRIVIAL SORT OF A 1-ELEMENT LIST
\}
\} if ( length(INSTANCE) == 1 )
dnode * Solution;
dnode * MERGEsort(dnode * INSTANCE)
particular activation of the (recursive) function MergeSort is doing.

Very Important: Automatic variables are used to keep track of what a
{ return Solution;

{ Solution = MERGE(Aus1, Aus2);
  node * Aus2 = MERGE2(Suppproblem2);
  node * Aus1 = MERGE2(Suppproblem1);
  /\RECURSE one or more times:\n
  Suppproblem2 = INSTANCE; //SUPPII removed half the original list.
  Suppproblem1 = SPLIT(INSTANCE);
  node * Suppproblem2, Suppproblem1;

} else {

  Solution = INSTANCE; //TRIVIAL SORT OF a 1-ITEM LIST

} if (LENGTH(INSTANCE) == 1)

{ node * MERGE2(node * INSTANCE)

University at Albany Computer Science Dept
particular activation of the (recursive) function MergeSort is doing.

VERY IMPORTANT: AUTOMATIC VARIABLES are used to keep track of what a
Recursive function.

Lab 4 is to observe the stack of activation records during the run of a

(insert into a stack is called push, delete from a stack is called pop).

and non-recursive.

other data relevant to all C/C++ function calls and returns, both recursive
runs C/C++ programs) Implementing and organizing local variables and

3. The "run-time stack" of activation records internal to the system when it

2. Storing and organizing intermediate results when evaluating expressions.

parenthesized expressions.

1. Figuring which pairs of parentheses MATCH in a correctly nested

3 uses for stacks:

ONLY ONE END (called the top).

that access, insertion and deletion are permitted at

What is a stack? A stack is a sequence that is restricted so
(a) One root node.
(b) Zero or more rooted trees, with no nodes or arcs in common with each other.

A rooted tree is a structure of nodes and arcs (pairs of nodes) that has:

What is a tree?
An expression: \( \text{(p)} \).

Are substrings of the expression:

\( \text{(e)} \) Any operator and operands under

\( \text{(d)} \) intersect

\( \text{(c)} \) One arc from this tree's root to

\( \text{(b)} \) Or has a top level operator, except

\( \text{(a)} \) Either is an identity or constant,

Under \( \text{(q)} \),

The root of each of the trees specified

\( \text{(r)} \) One arc from the root

\( \text{(s)} \) Or has more rooted trees, with

\( \text{(t)} \) One root node, and

A tree has:
XXX employees.

simultaneously with the complex C/C++ precedence/associativity rules' FIRE that

obvious. (2) If a programmer you are supervising likes to show off his/her

2 Practical Rules: (1) If it's doubtful or subtle, USE PARENTHESES to make it

\[
\begin{align*}
4 & = 2 \times \frac{3}{2} \\
9 & = 3 \times (3 + 4) \\
20 & = 3 \times (2 + 3)
\end{align*}
\]

not precedence than addition" from elementary school:

must memorize or look up. They begin with "multiplication has higher

Rather complicated operator precedence and associativity rules people

How do you know which operator is evaluated first?

Not fully parenthesized:

expressions.

To make learning these ideas easier, we will start with fully parenthesized
The top level operation Assign to A is executed LAST. Why MUST it be

7. Assign it also to A.

6. Assign the last sum to B.

5. Add subtraction's result to this last product.

4. Multiply F and G.

3. Subtract that from G, remember result.

2. Multiply old value of E by D.

1. Increment E first.

means:

$(((((F \times G) + ((E+D) \times C)) - B) = C) - D \times E + F \times G$
Example of an expression and its Parse Tree

```
(A = (B = ((C - (D * (E++))) + (F * G))))
```

(details cont'd on next 2 frames)

(contains on next 2 frames)
top level operator is multiplication

\((F \times G)\)
top level operator is increment (++)

(top level operator is multiplication (* ))

(top level operator is subtraction (-))

identifier
An expression (q).

1. Any operator and operands under
2. (and) overlaps!
3. Or more expressions as operands (no
4. If it has an operator, it has one
5. (and) sively, (and)
6. Or has a top level operator, except
7. Either is an identity or constant,
8. An expression:
9. Definitions: FIT these
10. FIT these
11. FIT these and expressions“ FIT these
12. Your job: Check that these examples, purporting
The root node: 

\(-((c-(d*(e++)))\))

Top level operator is subtraction (-)

Clause (d) is OK?

Is 0 or more which

other trees, nothing in common.

Tree de: clause (b) is OK?

Clause (e) is OK!
(a) An expression:

(b) (q)

(c) Any operator and operands under

(1) if it has an operator, it has one

(2) if it is an identity or constant,

and

no more expressions

and

silently.

(q) If it has a top level operator, except:

(1) Either is an identity or constant,

under (q).

The root of each of the trees specified

(1) one arc from this tree's root to

other or the root.

(2) Zero or more rooted trees, with

(1) one root node.

A tree has:

Definitions:

"trees and expressions" FIT these

Your job: check that these examples, pursuant
Expression de. Clause (b) is OK. 

The operands are substitions:

Expression de. Clause (c) is OK.

Identifier

++

Identifier

Expression de.

Has an operator:

(subtraction) (c - (d * (e++))

Expression de.

top level operator is multiplication ( * )

Identifier

++

Identifier

Expression de.

top level operator is increment ( ++ )

Identifier

Expression de.

top level operator is subtraction ( - )

Expression de.
and return its result.

(3) Combine the results from (2) using the meaning of the operator to compute

\[ \text{Evaluate(}L_1\text{)} \] call \text{Evaluate(}L_2\text{)} for each of the trees

(2) \text{RECURSIVEELY CALL Evaluate(}L_1\text{)}; call \text{Evaluate(}L_2\text{)} for each of the operands. (Only one call for a unary operator.)

identifiers. So, return it or its value.

(1) If \( L \) is just one node only, then the expression must be a constant or

Evaluate(\text{ParseTree } L)
\[ 4 - (6 - 4) \times \frac{3}{6+9} \]
\[(\langle 6-4 \rangle \ast (6+9)/(3/(6+9))) = 5/15 = 1\]

\[6 + 9 = 15\]

\[\langle 6-4 \rangle \ast (6+9)/(3/(6+9))\]
\[
\frac{15}{3} = 5
\]

\[
2 = 6 - 4
\]

\[
(\langle 4 \rangle \ast \frac{3}{3/(6+9)})
\]

\[
\frac{15}{3} = 5
\]

\[
6 + 9
\]

\[
\langle 4 \rangle - 6 - 4
\]

\[
\frac{15}{3} = 5
\]

\[
\langle 4 \rangle \ast \frac{3}{3/(6+9)}
\]

\[
6 + 9
\]

\[
\langle 4 \rangle - 6 - 4
\]

\[
\frac{15}{3} = 5
\]

\[
\langle 4 \rangle \ast \frac{3}{3/(6+9)}
\]
\((6+9)/3\) \[ \times \] \(6-4\) \[ = 15/3 \] \[ = 5 \]

\[10 = 5 \times 2 \text{ DONE!}\]
Combining is called “Mergeing". Combine the two sorted groups into one large sorted list, and return it. 

**INDEPENDENTLY?**

3. Sort each of these smaller groups (by recursive calls). That means size. "Split" in project 3.

2. Divide the elements to be sorted into two groups of equal (or almost equal).

1. If the input list has length 0 or 1, return it ("base case").

Arrays, which is easier?)

But you will implement merge sort on linked lists, not the beginning of DS0 13.2. But the problem of sorting a sequence (read the paradigm or pattern is applied to the problem of sorting a sequence when the divide-and-conquer MergeSort is the algorithm that is invented when the divide-and-conquer...
sorts a list of 7 letters.

Let's review the operation of the top level activation of MergeSort when it

University at Albany Computer Science Dept.
The two recursions have FINISHED; the top level activation will now merge.

Input

(G D A F C E B)

(G D A F)

(ADFG)

(BCE)

Recurse.

Recurse.

Recurse.

Recurse.

MERGE

SPLIT

SPLIT

MERGE

MERGE
Now, let's examine everything that happens during all the recursions.

Computation or activation was FINISHED.

Visual language convention: The indicated means "ghost": The indicated

BROWN
I plus the number of times you have to divide \( N \) by two with rounding to get

\[ \log(N) + 1 \]

The number of levels is rounded \( \log(N) + 1 \).

**Level 0:** 7 letters involved. ABCDEFG

**Level 1:** 7 letters involved. VAF D G

**Level 2:** 7 letters involved. AB C E B

**Level 3:** 7 letters involved. (ADF G)(B C E)

- The total number of elements involved in each level is \( N \) (the original input)
- The time used for recursions
- This time OMITS the time used for recursions
- Operations is LESS THAN (constant) * Number of elements
- During any one single activation, the time used to SPLITT, MERGE and merge.

"All the work is done by MERGE."

**Why is MergeSort so Fast?**