Merge Sort of Linked Lists
Trees, Recursion, Expressions, Stacks
How Recursion Works.

CSI 310: Lecture 15
First, note that \( n \) is a local variable.

```c
int fact(int n)
{
    if (n == 0)
        return 1;
    else
        return fact(n-1) * n;
}
```
We now dramatize what happens when the computer executes:

\[
\begin{align*}
\text{\textbf{n is an AUTOMATIC VARIABLE in each of 3 functions}}. \\
& \quad \{ \text{/* CRASH } \} \text{ \textbf{else assert (0);}} \\
& \quad \{ \text{else return Func(n-1);} \} \\
& \quad \{ \text{else return Func(n-1);} \} \\
& \quad \{ \text{else return Func(n-1);} \} \\
& \quad \{ \text{int Func(int n) \{ if (n==0) return 1; int Func(int n) \{ if (n==0) return 1; int Func(int n) \{ if (n==0) return 1; int Func(int n) \{ if (n==0) return 1; \}} \}} \}} \}
\end{align*}
\]
Different activations—different automatic variables

The storage 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.
Let's do it again with a recursive function:

```c
int fact(int n) {
    if (n == 0) return 1;
    else return fact(n-1) * n;
}

3. fact(5)
2. fact(2)
1. fact(0)
```
4. Now, automatic variables are in the activation record of the activation.

RETURNED TO.

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

2. This activation's activation record
   I. The return value (if any) is saved for use by the caller.

When a function activation

execute the return statement:

RETURN

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
But, running \texttt{factor(0)} calculates \texttt{0==0} is true and returns \texttt{1}.

For example, running \texttt{factor(720)} computes \texttt{720 \ast 7 = 5040} and returns \texttt{It}.

For example, running \texttt{factor(720)} computes \texttt{7 \ast 1 = 6}, calls \texttt{factor(6)}, which returns

\begin{verbatim}
if (n==0) return 1;
else return factor(n-1)*n;
\end{verbatim}

\texttt{int fac(int n)}

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

\textbf{Definition:} A function is \textbf{recursive} means the body of the function sometimes,

\begin{verbatim} re-implemented \texttt{if \textit{it}'s worth it.}
\end{verbatim}

(3) \textbf{Elegant way to write programs; performance can be improved with routine}

(2) \textbf{Powerful problem solving technique.}

(1) \textbf{Understanding, not just programming, data structures and algorithms.}

Recursion: Trees, Recursion, Expressions, Stacks are closely related.
How you observed this to turnin-cs1310 - c cs1310 - p tab4 recursive function variables. For a checkoff, demonstrate or submit an essay on observe that DIFFERENT activation records have DIFFERENT copies of local observe that perhaps within the help of the lab instructor and/or classmates, how to figure out.

Copy, build, and run it under the debugger. You will find an expanded version of the "vertical" program in ~acs1310/lab4.

Read section 9.1 of the text and bring your textbook to the lab.

Exercise: Lab4
{  
    return Solution;

    Solution = result from COMBINING CALCULATION (Ans1, Ans2, etc.);
    answer-type Ans2 = SolutionFun(Subproblem2); etc.;
    answer-type Ans1 = SolutionFun(Subproblem1);
    // RECURSE one or more times:
    // BREAK up instance into Subproblem1, Subproblem2, etc.
    input-type Subproblem1, Subproblem2, etc.;
}

else
    {
        Solution = result from DIRECT CALCULATION (instance);
    }

    if (instance is small enough)
    {  
        answer-type Solution;
        answer-type SolutionFun(input-type instance)
    }

    Divide and Conquer Pattern

University at Albany Computer Science Dept.
{ return Solution; }

{ Solution = MERGE( Ans1, Ans2 );
  node * Ans2 = MERGESORT( Subproblems2 );
  node * Ans1 = MERGESORT( Subproblems1 );
  node * Ans1 = node * Subproblems1, Subproblems2;
  if (Length(Instance) == 1)
    node * Solution = TRIVIAL SORT OF a 1-Element Instance;
  else
    Solution = Instance; /*TRIVIAL SORT OF a 1-Element Instance */
  }

Divide and Conquer Pattern Applied to Sorting
of the half that WAS REMOVED from the linked list.

(2) The return value is the addr. of the first node
the nodes removed.

halt

[Instance=sortedList.value, but with approx. half]

linked list of AT LEAST TWO C-strings.

Pre: Instance=the addr. of the first node of a
node * Split (node * Instance)

C-strings will be compared using strcmp().

node is a linked-list node type; each node's data is the address of a C-string.

Merge sort explained using Pre/Postconditions:
NOT SORTED AT ALL.
The pre-condition for MergeSort means the input is for Merge means most sorting work was already finished.
The post-conditions are identical. But, the pre-condition

```c
strcmp() order.
output the linked list of the original C-strings sorted into
Post: The return value is the addr. of the first node
distinct, already sorted lists of C-strings.
pre: List1 and List2 both double addresses of first nodes of

dnode * MergeSort(dnode * List1, dnode * List2);
non-functional Requirement: Use the Merge Sort Algorithm.

```
```c

```c
non-empty) linked list of C-strings.
pre: Instance=the addr. of the first node of a

dnode * MergeSort(dnode * Instance);

```
{ return Solution;

 Solution = MERGE (Ans1, Ans2); 
 do while Ans2 = MERGEsort (Subproblem2)
 do while Ans1 = MERGEsort (Subproblem1)
 RECURSE one or more times:
 Subproblem2 = Instance; //SPLIT removed half the original list.
 Subproblem1 = SPLIT (Instance);
 do while Subproblem1, Subproblem2
 }
 else
 { Solution = Instance; //TRIVIAL SORT OF a 1-element list
 }
 if (length (Instance) == 1)
 do else Solution;
 do while MERGEsort (do while Instance)
Subpr1 = SPLIT(Instance); Subpr2 = Instance

Subpr1

Subpr2

Instance
I Don't Care about ANYTHING ELSE

What is the result of executing this line of code?

\[ \text{node * Ans1} = \text{MergeSort ( Subpr1 )} \]
dnode * Ans1 = MergeSort( Subpr1 );

What is the result of executing this line of code?

dnode * Ans1 = MergeSort( Subpr1 );
MergeSort(Subpr2)!

Ans2 =

What is the result of executing:

Subpr2
The result of the recursive call is to make the returned list sorted properly! The result of

\[
\text{ans2} = \text{mergeSort}(\text{Subpr2})
\]
Let's complete the job: First, redraw the diagram...
MERGE
Combination operation:

```
MergeSort(Subpr1);
Ans1 = MergeSort(Subpr2);
Ans2 = node * Ans1 = MergeSort(Subpr2);
```

These two sorted lists...
move all remaining list1 or list2 entries to end of outputlist;

// end of while loop

{ // move the first entry from list2 to end of outputlist;

} // move the first entry from list1 to end of outputlist;

( the first entry in list2 is before or equal to the first entry in list1 )

there is a first entry in list2

else ( there is a first entry in list1 )

outputlist = empty list;

MERGE ( sorted list1, sorted list2 ) algorithm:
(p) One arc from this tree's root to the root of each of the trees specified under or the root. (and)

(c) Zero or more rooted trees, with no nodes or arcs in common with each other.

(4) One root node. (and)

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

What is a tree?
An expression

(q) are substrings of the expression.

Any operator and operands under

overlapped (and) no more expressions

If it has an operator, it has one

sively (and) or has a top level operator, excru-

Either is an identifier or constant.

A tree has

the root of each of the trees specified

(c) One arc from this tree's root to

other or the root (and)

no nodes or arcs in common with each

(q) zero or more rooted trees with

(a) One root node (and)

University at Albany Computer Science Dept.
XXX employee

smartness 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

\[
\begin{align*}
35 & = 5 \times 7 = 5 \times (4 + 3) \\
30 & = 3 + 20 = 3 \times 4 + 3 \\
23 & = 3 \times 4 + 3 + 6 \\
\end{align*}
\]

not precedence than addition. From elementary school: multiplication has higher

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?

\[
A = B + C - D * E + F * G
\]

Not fully parenthesized:

expressions.

To make learning these ideas easier, we will start with fully parenthesized
done last? It uses the results of the all previous operation!

The top level operation "Assign to A" is executed first. 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:

\[
A = (B) = (C) - (D) * (E++) + (F) * (G)
\]

Fully parenthesized:

\[
A = B = C - D * E++ + F * G
\]
Example of an expression and its Parse Tree

\[(A = (B = ((C - (D \times (E++))) + (F \times G))))\]
top level operator is multiplication

(F*G)
(C-(D*(E++)))

- Top level operator is subtraction (-)

  (D*(E++))

- Top level operator is multiplication (*)

  (E++)

  top level operator is increment (++)

  E

  Identifier

- Top level operator is multiplication (*)

  (D*(E++))

  top level operator is increment (++)

  D

  Identifier

- Top level operator is subtraction (-)

  (C-D*(E++))

  top level operator is multiplication (*)

  (D*(E++))

  top level operator is increment (++)

  C

  Identifier
(p). An expression is a subject of the expression.

(c) Any operator and operands under

and overlap(s).

or more expressions as operands (no

operator, it has one

operator, it has none

operator, it has none

operator, it has none)

(s) Either is an identifier or constant,

An expression:

definitions:

trees and expressions. Fill these

Your job: Check that these examples, purporting

trees and expressions
Is 0 or more which

identifier

(identifier)

C

D

E

++

* (identifier)

( D*(E++) )

++

- (C-(D*(E+++)))

Tree der. clause (a) is O.K.

Clause (c) is O.K. too?

other trees, nothing in common.

Tree der. clause (b) is O.K.

Which

The root node:
The operands are substitution:

Expression def. Clause (p) is OK.

Expression def. Clause (c) is OK.

Not overlapping.

Expressions as operands.

Is 1 or more

Which

Clause (a)

Clause (b)

Has an operator:

Identifier

Identifier

Identifier

Identifier

Top level operator is increment (+)

Top level operator is multiplication (*)

Top level operator is subtraction (−)

(C−(D*(E++)))

(D*(E++))
and return its result.

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

(2) RECURSIVELY Call Evaluate(T1), call Evaluate(T2) for each of the trees

identifier. So, return its value.

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

Evaluate(ParseTree T)

tree:

The following recursive algorithm evaluates an expression when given its parse.

The free of an expression is called the expression's Parse Tree. Parsing expression is called free of the expression. The (rather difficult and non-trivial) job of figuring out the tree from a given absolute clarity.

The tree of an expression represents the expression's structure with

University at Albany Computer Science Dept.
A stack is a sequence that is restricted so that access, insertion and deletion are permitted at only one end (called the top). What is a stack? A stack is a sequence that is restricted so that access, insertion and deletion are permitted at only one end (called the top).

2. Storing and organizing intermediate results when evaluating expressions.

3. Organizing which pairs of parentheses MATCH in a correctly nested parenthesized expression.

3. (The "run-time stack" of activation records, internal to the system when it runs C/C++ programs.) Implementing and organizing local variables and other data relevant to all C/C++ function calls and returns, both recursive and non-recursive.

Inside a stack is called push, delete from a stack is called pop.

To observe the stack of activation records during the run of a recursive function.
or composed of its individual variables, taken together.

It is useful to consider the WHOLE ARRAY as ONE VARIABLE that is formed.

C/C++ the indexes range from 0 to Length-1.

- Each element is selected for access using an integer, called an index. In

- Each element is selected for access using an integer, called an index. In

addressess, like a row of houses on one city block.

- The elements are located contiguously in memory, at adjacent

- The number of elements (Length of the array) is fixed.

- char, any other type...

- Each individual variable, called an element, has the same type (int, float, 

An array is a sequence of variables (plural) that:

Arrays, again.
int i;

cout << Type vector M:


cout << Type vector V:

double & V[1][3], W[3][3], sum = 0;

to add vector V and W:

Mathematical vectors e.g. coordinates of points in 3 dimensions. C++ code

Depending on how you tell WHERE THE END IS.

... holds strings up to 99 or 100 chars long

char myString[100];

char string[] = "Some string data. This is called string data."

What can you use arrays for?

University at Albany Computer Science Dept.
This prints each number and price, separated by 6 spaces:

```c
for (i = 0; i < 100; i++)
    count >> printf("%d \t", stock[i].price);
```

Get prices of stocks numbered 0 to 99:

```
float price[100];
```

Statistics, such as prices of 100 different stocks:

```c
    { [i]sum += count } (++i; 0 <= i < 3); for
    { [i]M << count } (++i; 0 <= i < 3); for
    { [i]M + [i]A = [i]sum } (++i; 0 <= i < 3); for
    { [i]M << count } (++i; 0 <= i < 3); for
```
\[C\text{-strings are different from } C++ \text{ strings you get from } \#include \text{ \texttt{<string>}}\]

\[
\text{\texttt{W}}[0] = \text{\texttt{A}}, \quad \text{\texttt{W}}[1] = \text{\texttt{B}}, \quad \text{\texttt{W}}[2] = \text{\texttt{C}}, \quad \text{\texttt{W}}[3] = \text{\texttt{D}}.
\]

The C-string "ABCD" (4 letters) is stored in a \texttt{LENGTH} of 5 (give, not 4) char

Strings in char arrays terminated with \texttt{\textbackslash 0}, are called C-strings

The null char is coded \texttt{\textbackslash 0},

In C/C++ the char "A" is coded \texttt{\textbackslash 41},

called the "null char".

One way to tell where the end of a string is: Just after the last element used for
cout >> mychararray;  // Print the value you typed:
cin >> mychararray;  // Get it from stdin.

Reading up to 11 characters you type on one input line:
// Holds a c-string with length up to 11
char mychararray[12];  // REQUIRED in CSt10: Declare a variable that can hold a c-string:

cout >> "Hello World";  // Got it from stdin. You have used them in CSt20!
C-strings are very easy to use. You have used them in CSt20!

#include <iostream>

using namespace std;

University at Albany Computer Science Dept.
{ return 0;
}

while (!finished); //*
    cout << A[i] << endl; // A sorted string was printed.
} now, A[0..nch-1] is sorted

for( j = i+1; j < nch; j++)
for( i=0; i < nch-1; i++)
    ( ( [j] < [i] )

while( cin >> tsize) //
    ( cin >> tsize = 100; int nch, j; char A[tsize];
using namespace std;

#include <cstdlib>
#include <iostream>

Selection sort demo: processes chars within the array A.
The other half is to implement the recursive MergeSort algorithm, a topic for next week, after the midterm.

We now illustrate what half your Project 3 work must do.

We sort the lines in the Project 2 "main list" lexicographically, as C-strings, sort the integers in the Project 3 consists of implementing two sorting algorithms and applying them to two different types of data structures. It uses the Selection Sort algorithm for sorting.

This program manipulates characters as if they were numbers.
Sample list of items to sort:
How can we very efficiently swap the strings in the nodes pointed to by $i$ and $j$?
Computer DOES NOT copy chars nor node pointers!

Swap the values in the 2 data fields of the nodes pointed to by I and J.
and finally d.

prints the characters in order. H', then e, two I's, an o, a space, then W', o, I, T,' count >> "Hello World"

is easier to think about than prints the string Hello World
count >> "Hello World"

C-string:

array of char (sequence of char variables) is a single variable that holds ONE

The examples of simple C-string use illustrate the usefulness of thinking that an
return 0;
} // end of main()
function

cout >> "Input from cin failed. exiting" >> endl;

{ // process the input somehow...
    ...

    if (DEBUG) { cout >> input >> endl; }

    while (cin >> getline(input, INBUF_SIZE)) {
        char input[INBUF_SIZE]
        const int INBUF_SIZE = 12;
    }

    main()
Recognizing one-line commands:

University at Albany Computer Science Dept.
// end of main() function.
return 0;

} // cout << "Input from cin failed. exiting"} end;
}