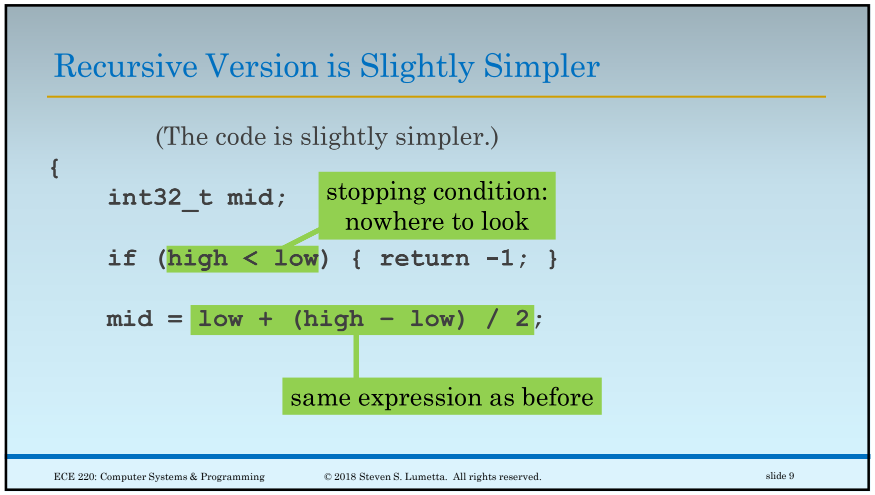


Recursive Version is Slightly Simpler

(The code is slightly simpler.)

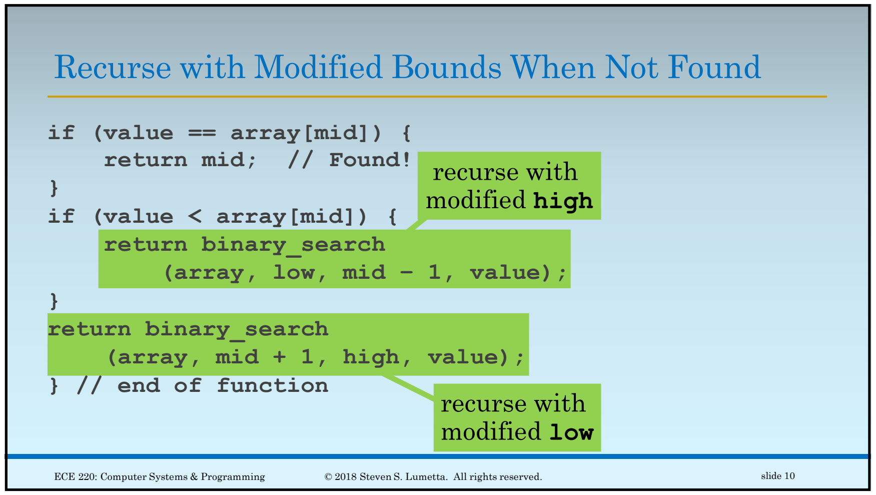
```
{
    int32_t mid;

    if (high < low) { return -1; }

    mid = low + (high - low) / 2;
```

stopping condition: nowhere to look

same expression as before

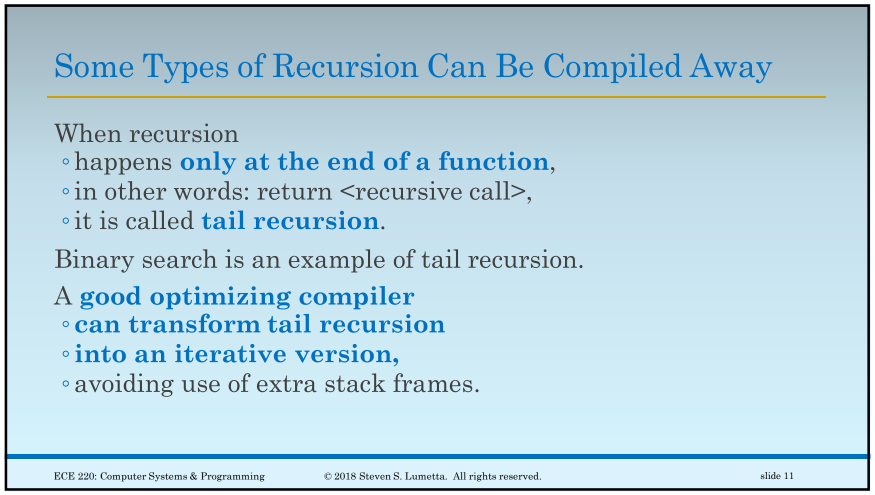


Recurse with Modified Bounds When Not Found

```
if (value == array[mid]) {
    return mid;   // Found!
}
if (value < array[mid]) {
    return binary_search
        (array, low, mid - 1, value);
}
return binary_search
    (array, mid + 1, high, value);
} // end of function
```

recurse with modified **high**

recurse with modified **low**

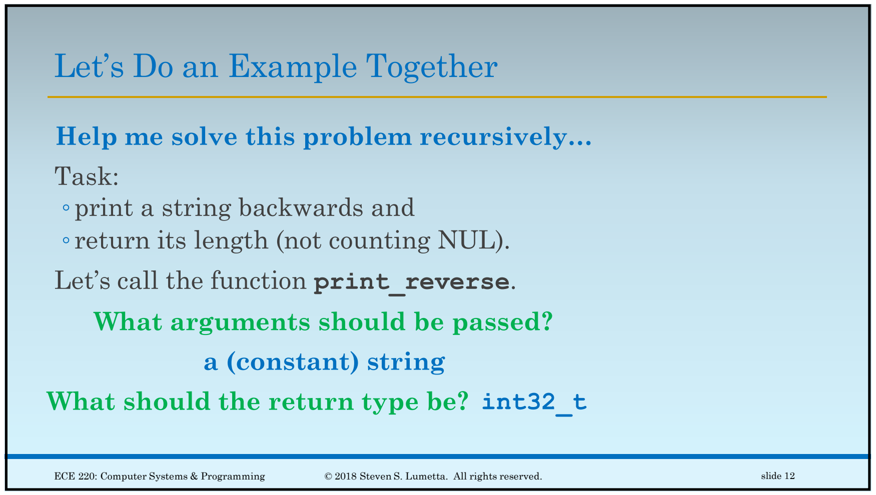


Some Types of Recursion Can Be Compiled Away

When recursion
- happens **only at the end of a function**,
- in other words: return <recursive call>,
- it is called **tail recursion**.

Binary search is an example of tail recursion.

A **good optimizing compiler**
- **can transform tail recursion**
- **into an iterative version,**
- avoiding use of extra stack frames.



Let's Do an Example Together

Help me solve this problem recursively…

Task:
- print a string backwards and
- return its length (not counting NUL).

Let's call the function **`print_reverse`**.

What arguments should be passed?

a (constant) string

What should the return type be? `int32_t`