CS 334
lecture 11
Dynamic Scope

Follow control links until variable is found.

Static Scope

- Need to find memory for var declared in the closest lexically enclosing scope.

- Based on program text.
Access Link

1) Pointer to the A.R. corresponding to the lexically enclosing block of code.

2) Compiler computes # of access links to follow to find var based on source code.

Static vs. Dynamic Scope

+ easier to reason about code.
+ less danger of accidentally redefining a name.
Closure - Functions Passed as args.

env. ptr: points to A.R. for the functions definition.
- Functions returned from Functions

- Can't use stack for A.Rs.

- Use heap!
Tail Recursion

sumSquares \( O(n) \) space.

val v = sumSquares Tail (2, 0)
Tail Recursion Optimization

- Guy Steele

```latex
\text{fun sumSgLoop}(n, acc) \{
    \text{while } (\text{not } (n \leq 0)) \{
        acc = \text{n} \times n + \text{acc};
        n = n - 1;
    \}
    \text{return acc;}
}\)
```

\[O(n) \text{ space}\]

\[\rightarrow O(1) \text{ space.}\]
Reverse

$O(n)$ space
$O(n^2)$ time.

<table>
<thead>
<tr>
<th>list</th>
<th>acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1, 2, 3]</td>
<td>[3]</td>
</tr>
<tr>
<td>[2, 3]</td>
<td>[1]</td>
</tr>
<tr>
<td>[3]</td>
<td>[2, 1]</td>
</tr>
<tr>
<td>[]</td>
<td>[3, 2, 1]</td>
</tr>
</tbody>
</table>

Tail recursive:

$O(1)$ space
$O(n)$ time.