CSC180: Lecture 14

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgement: These slides are partially based on the slides supplied with Prof. Savitch book: Problem Solving with C

Recursion

Recursive Functions

- A recursive function contains a call to itself
- When breaking a task into subtasks, it may be that the subtask is a smaller example of the same task
  - Searching an array could be divided into searching the first and second halves of the array
  - Searching each half is a smaller version of searching the whole array
  - Tasks like this can be solved with recursive functions

A Closer Look at Recursion

- Recursive calls are tracked by
  - Temporarily stopping execution at the recursive call
    - The result of the call is needed before proceeding
  - Saving information to continue execution later
  - Evaluating the recursive call
  - Resuming the stopped execution

How Recursion Ends

- Eventually one of the recursive calls must not depend on another recursive call
- Recursive functions are defined as
  - One or more cases where the task is accomplished by using recursive calls to do a smaller version of the task
  - One or more cases where the task is accomplished without the use of any recursive calls
    - These are called base cases or stopping cases

"Infinite" Recursion

- A function that never reaches a base case, in theory, will run forever
  - In practice, the computer will run out of resources and the program will terminate abnormally

Example: Infinite Recursion

- Function write_vertical, without the base case

```
void new_write_vertical(int n)
    {
        new_write_vertical (n /10);
        printf( “%d \r\n”, n % 10 );
     }
```

will eventually call write_vertical(0), which will call write_vertical(0),which will call write_vertical(0), which will call write_vertical(0), which will call write_vertical(0), which will call write_vertical(0), which will call write_vertical (0), …

Program Example:
A Powers Function

$2^3 = 8$

$2 * 2 * 2$

$9^2 = 81$

Program Example: A Powers Function

- To define a new power function that returns an int, such that

  int y = power(2,3);

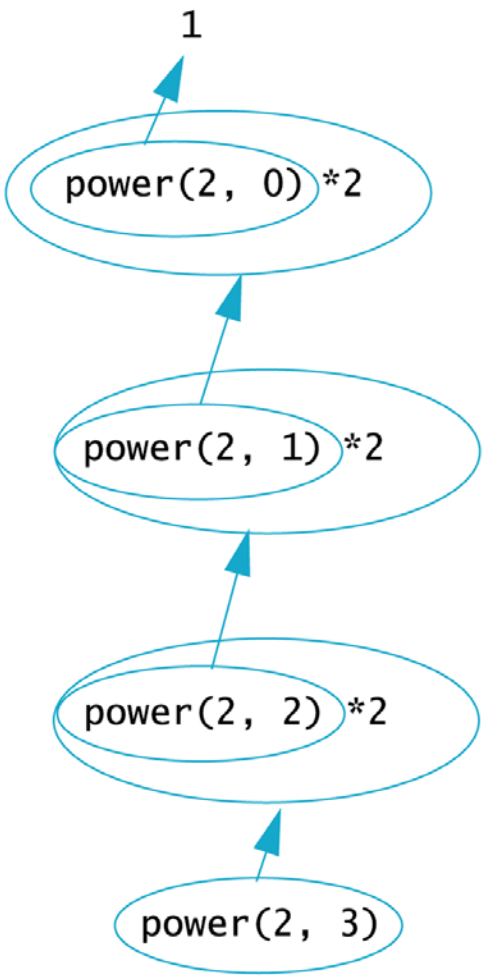
  places 23 in y
  - Use this definition:

    xn = xn-1 * x
  - Translating the right side to C++ gives:

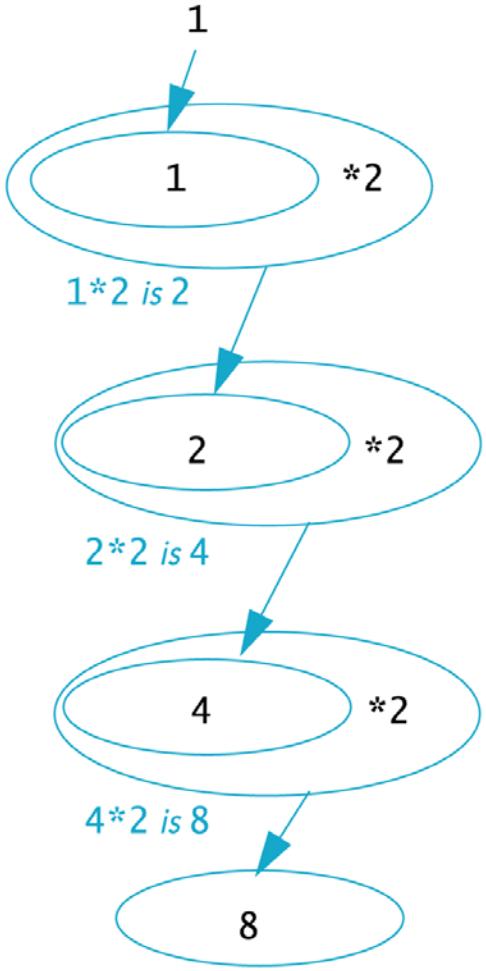
    power(x, n-1) * x
  - The base case: n = = 0 and power should return 1

power(2, 3)

- Rethinking Power(2, 3) …. 2* 2 * 2
  - power(2, 3) is power(2, 2) * 2
  - Power(2, 2) is power(2, 1) * 2
  - Power(2, 1) is power(2, 0) * 2
  - Power (2, 0) is 1



Start Here



power(2, 3) is 8