CSE 143
Lecture 13

Recursive Programming

reading: 12.2 - 12.3

slides created by Marty Stepp
http://www.cs.washington.edu/143/
Exercise

- Write a recursive method `pow` accepts an integer base and exponent and returns the base raised to that exponent.
  - Example: `pow(3, 4)` returns 81

- Solve the problem recursively and without using loops.
/ Returns base ^ exponent.  
// Precondition: exponent >= 0  
public static int pow(int base, int exponent) {  
    if (exponent == 0) {  
        // base case; any number to 0th power is 1  
        return 1;  
    } else {  
        // recursive case:  x^y = x * x^(y-1)  
        return base * pow(base, exponent - 1);  
    }  
}  

• Notice the following mathematical property:

\[ 3^{12} = 531441 = 9^6 \]

\[ = (3^2)^6 \]

\[ 531441 = (9^2)^3 \]

\[ = ((3^2)^2)^3 \]

– When does this "trick" work?

– How can we incorporate this optimization into our pow method?

– What is the benefit of this trick if the method already works?
/\ Returns base ^ exponent.
/\ Precondition: exponent >= 0

public static int pow(int base, int exponent) {
    if (exponent == 0) {
        // base case; any number to 0th power is 1
        return 1;
    }
    else if (exponent % 2 == 0) {
        // recursive case 1: x^y = (x^2)^(y/2)
        return pow(base * base, exponent / 2);
    }
    else {
        // recursive case 2: x^y = x * x^(y-1)
        return base * pow(base, exponent - 1);
    }
}
Exercise

- Write a recursive method `printBinary` that accepts an integer and prints that number's representation in binary (base 2).
  
  - Example: `printBinary(7)` prints 111
  - Example: `printBinary(12)` prints 1100
  - Example: `printBinary(42)` prints 101010

  
  - Write the method recursively and without using any loops.
Case analysis

• Recursion is about solving a small piece of a large problem.
  – What is 69743 in binary?
    • Do we know *anything* about its representation in binary?
  
  – Case analysis:
    • What is/are easy numbers to print in binary?
    • Can we express a larger number in terms of a smaller number(s)?

  – Suppose we are examining some arbitrary integer \( N \).
    • if \( N \)'s binary representation is 10010101011
    • \( (N \div 2) \)'s binary representation is 1001010101
    • \( (N \mod 2) \)'s binary representation is 1
printBinary solution

// Prints the given integer's binary representation.
// Precondition: n >= 0
public static void printBinary(int n) {
    if (n < 2) {
        // base case; same as base 10
        System.out.println(n);
    } else {
        // recursive case; break number apart
        printBinary(n / 2);
        printBinary(n % 2);
    }
}

– Can we eliminate the precondition and deal with negatives?
// Prints the given integer's binary representation.
public static void printBinary(int n) {
    if (n < 0) {
        // recursive case for negative numbers
        System.out.print("-");
        printBinary(-n);
    } else if (n < 2) {
        // base case; same as base 10
        System.out.println(n);
    } else {
        // recursive case; break number apart
        printBinary(n / 2);
        printBinary(n % 2);
    }
}
Exercise

• Write a recursive method `isPalindrome` accepts a `String` and returns `true` if it reads the same forwards as backwards.

- `isPalindrome("madam")` → true
- `isPalindrome("racecar")` → true
- `isPalindrome("step on no pets")` → true
- `isPalindrome("able was I ere I saw elba")` → true
- `isPalindrome("Java")` → false
- `isPalindrome("rotater")` → false
- `isPalindrome("byebye")` → false
- `isPalindrome("notion")` → false
// Returns true if the given string reads the same
// forwards as backwards.
// Trivially true for empty or 1-letter strings.
public static boolean isPalindrome(String s) {
    if (s.length() < 2) {
        return true;  // base case
    } else {
        char first = s.charAt(0);
        char last  = s.charAt(s.length() - 1);
        if (first != last) {
            return false;
        }  // recursive case
    String middle = s.substring(1, s.length() - 1);
    return isPalindrome(middle);
}
}
// Returns true if the given string reads the same
// forwards as backwards.
// Trivially true for empty or 1-letter strings.
public static boolean isPalindrome(String s) {
    if (s.length() < 2) {
        return true;    // base case
    } else {
        return s.charAt(0) == s.charAt(s.length() - 1)
            && isPalindrome(s.substring(1, s.length() - 1));
    }
}
Exercise

• Write a recursive method `reverseLines` that accepts a file `Scanner` and prints the lines of the file in reverse order.

  – Example input file:
    
    Roses are red,  
    Violets are blue.  
    All my base  
    Are belong to you.
  
  – Expected console output:
    
    Are belong to you.  
    All my base  
    Violets are blue.  
    Roses are red,

  – What are the cases to consider?
    
    • How can we solve a small part of the problem at a time?
    • What is a file that is very easy to reverse?
Reversal pseudocode

• Reversing the lines of a file:
  – Read a line L from the file.
  – Print the rest of the lines in reverse order.
  – Print the line L.

• If only we had a way to reverse the rest of the lines of the file....
Reversal solution

public static void reverseLines(Scanner input) {
    if (input.hasNextLine()) {
        // recursive case
        String line = input.nextLine();
        reverseLines(input);
        System.out.println(line);
    }
}

- Where is the base case?
Tracing our algorithm

- **call stack**: The method invocations running at any one time.

```java
public static void reverseLines(Scanner input) {
    if (input.hasNextLine()) {
        String line = input.nextLine();  // "Roses are red,"
        reverseLines(input);
        System.out.println(line);
    }
}

public static void reverseLines(Scanner input) {
    if (input.hasNextLine()) {
        String line = input.nextLine();  // "Violets are blue."
        reverseLines(input);
        System.out.println(line);
    }
}

public static void reverseLines(Scanner input) {
    if (input.hasNextLine()) {
        String line = input.nextLine();  // "All my base"
        reverseLines(input);
        System.out.println(line);
    }
}

public static void reverseLines(Scanner input) {
    if (input.hasNextLine()) {
        String line = input.nextLine();  // "Are belong to you."
        reverseLines(input);
        System.out.println(line);
    }
    // false
    ...
}
```

Roses are red,
Violets are blue.
All my base
Are belong to you.