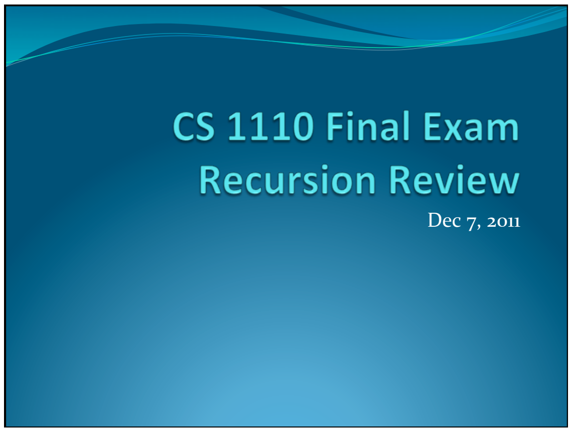
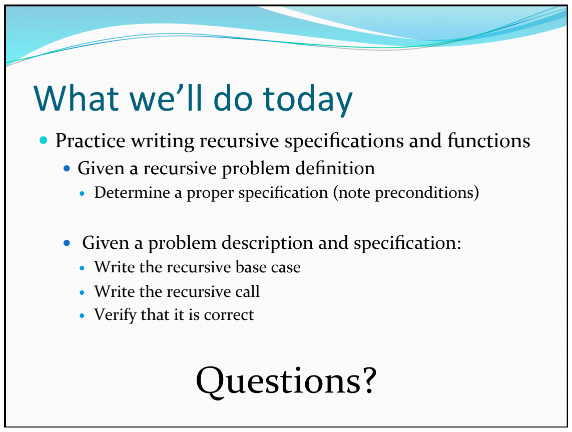


CS 1110 Final Exam Recursion Review

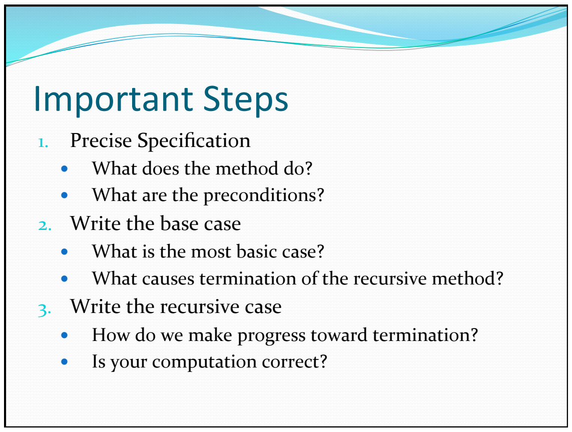
Dec 7, 2011



What we'll do today

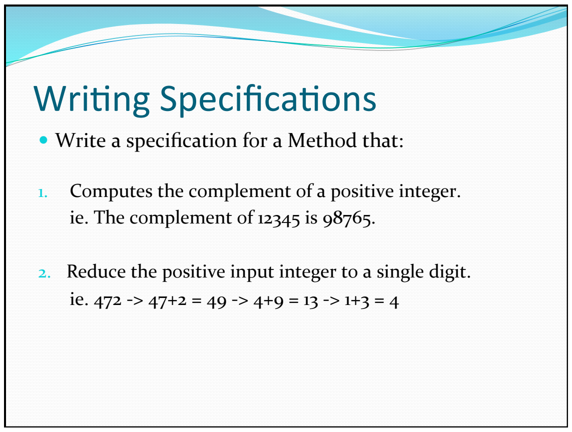
- Practice writing recursive specifications and functions
  - Given a recursive problem definition
    - Determine a proper specification (note preconditions)
  - Given a problem description and specification:
    - Write the recursive base case
    - Write the recursive call
    - Verify that it is correct

Questions?



Important Steps

1. Precise Specification
   - What does the method do?
   - What are the preconditions?
2. Write the base case
   - What is the most basic case?
   - What causes termination of the recursive method?
3. Write the recursive case
   - How do we make progress toward termination?
   - Is your computation correct?

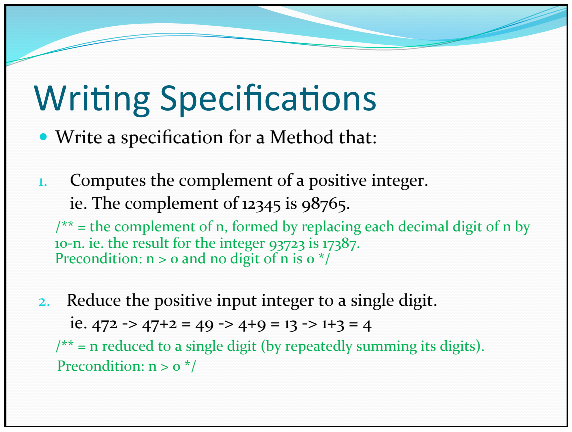


Writing Specifications

- Write a specification for a Method that:

1. Computes the complement of a positive integer.
   ie. The complement of 12345 is 98765.

2. Reduce the positive input integer to a single digit.
   ie. 472 -> 47+2 = 49 -> 4+9 = 13 -> 1+3 = 4

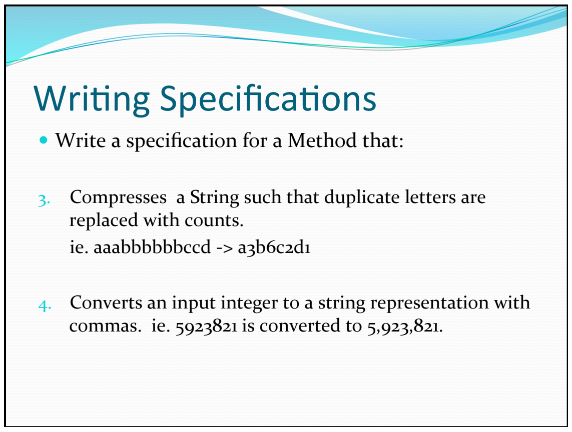


Writing Specifications

- Write a specification for a Method that:

1. Computes the complement of a positive integer.
   ie. The complement of 12345 is 98765.
   /** = the complement of n, formed by replacing each decimal digit of n by 10-n. ie. the result for the integer 93723 is 17387.
   Precondition: n > 0 and no digit of n is 0 */

2. Reduce the positive input integer to a single digit.
   ie. 472 -> 47+2 = 49 -> 4+9 = 13 -> 1+3 = 4
   /** = n reduced to a single digit (by repeatedly summing its digits).
   Precondition: n > 0 */



Writing Specifications

- Write a specification for a Method that:

3. Compresses a String such that duplicate letters are replaced with counts.
   ie. aaabbbbbbccd -> a3b6c2d1

4. Converts an input integer to a string representation with commas. ie. 5923821 is converted to 5,923,821.

Writing Specifications

- Write a specification for a Method that:

3. Compresses a String such that duplicate letters are replaced with counts.
   ie. aaabbbbbbccd -> a3b6c2d1

   /** = s compressed such that duplicates are replaced with the count of how many occurrences that character has in a row.*/

4. Converts an input integer to a string representation with commas. ie. 5923821 is converted to 5,923,821.

   /** = String representation of integer with commas added*/

Complement of an Integer

```
/** = the complement of n, formed by replacing
   each decimal digit of n by 10-n.
   ie. the result for the integer 93723 is 17387.
   Precondition: n > 0 and no digit of n is 0 */
   public static int complement(int n) {
      // Base Case

      // Recursive Case

   }
```

Complement of an Integer

```
/** = the complement of n, formed by replacing
   each decimal digit of n by 10-n.
   ie. the result for the integer 93723 is 17387.
   Precondition: n > 0 and no digit of n is 0 */
   public static int complement(int n) {
      // Base Case

      if (n < 10)
         return 10 - n;
      // Recursive Case

      return complement(n/10) * 10 + (10 - n%10);
}
```

Problem: Properly add commas to an integer and return the string representation. ie. 5923821 is converted to 5,923,821.

```
/** = String representation of integer with commas added*/
public static String addCommas(int n) {
   // Base case
   if (n < 1000)
      return “” + n;
   // Recursive Case
   String number = "" + n;
   return addCommas (n/1000) + ",“ +
         number.substring(number.length()-3);
}
```

Is something wrong?

Problem: Properly add commas to an integer and return the string representation. ie. 5923821 is converted to 5,923,821.

```
/** = String representation of integer with commas added*/
public static String addCommas(int n) {
   if (n < 0) return "-" + addCommasHelper(-n);
   else return addCommasHelper(n);
}
/** = String representation of a positive integer with commas added.
    Precondition: n >= 0*/
private static String addCommasHelper(int n) {
   // Base case
   if (n < 1000)
      return “” + n;
   // Recursive Case
   String number = "" + n;
   return addCommasHelper(n/1000) + ",“ + number.substring(number.length()-3);
}
```

An extra problem…

Given:
  Class FacebookProfile
      public String getName();
      public Vector<FacebookProfile> getFriends();

We want to answer the question:
  Is this FacebookProfile at most 6 degrees away from Kevin Bacon?

Specification:

/** = “this FacebookProfile is at most 6 degrees away from Kevin Bacon” */

6-Degrees of Kevin Bacon

```
/** = “this FacebookProfile is at most 6 degrees away from Kevin Bacon” */
public boolean sixDegreesOfKevinBacon() {
    return sixDegreesHelper(6);
}

/** = “this FacebookProfile is at most n degrees away from Kevin Bacon” */
private boolean sixDegreesHelper(int n) {
    // Base case
    if (getName().equals(“Kevin Bacon”))
        return true;
    if (n == 0)
        return false;
    // Recursive Case
    Vector<FacebookProfile> friends = getFriends();
    for (int i=0; i<friends.size(); i++) {
        if (friends.get(i).sixDegreesHelper(n-1))
            return true;
    }
    return false;
}
```

Extra Problems

- Given an array, user recursion to determine if it is sorted
- Given a String s, list all the permutations of String s:
  - “XZY” → “XYZ”, “XZY”, “ZYX”, “YXZ”, etc
- Use recursion to find the minimum element in an array

Questions?

Good Luck!

- Don’t Stress!
- Take your time!
- Have a great winter break!