1. True/False

T/F: Recursive functions should always return a value.

Answer: False, there is a legitimate example of a void function that is recursive that outputs a string backwards in the book. (Page 57)

1. Multiple Choice

Which of the following is a strong guideline (if not a rule) for solving recursive problems?

1. “*Recursion is most valuable when it can be used to replace iterative solutions*”
2. There should be more recursive calls than base cases in a recursive function
3. Tail recursion is an interesting concept, but should be avoided in practice
4. The problem will get smaller with each recursive call

Answer: D. A: False, recursive solutions should be replaced with iterative ones when the iterative solution is not too complex, B: False, this is not necessarily true; although, a function probably should feature more than one base case if it has more than one recursive call, C: False, tail recursion often makes for an easier to understand solution, (D): True, it is important that the problem gets smaller with each recursive call or there could be cases where the algorithm will never terminate. (Pages 86, 84, 87, 58)

1. Fill in the Blank

“A recursive solution that recomputes certain values frequently can be quite \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. In such cases, \_\_\_\_\_\_\_\_\_\_\_\_\_ may be preferable to recursion.”

Answer: inefficient; iteration. Synonyms should also be accepted. (Page 82)

1. Short Answer or Code

Write a tail recursive function that searches an integer array one element at a time until a certain value is found. The search should start at the last element and work towards the beginning. If the value is never found, use “-1” to signal that the element could not be found. The function should return the index in the array where the target value is encountered by the function.

(An) Answer:

int findByTailRecursion (int\* theArray, int lastIndex, int target) {

//define base cases

// case: the target value is found

if(theArray[lastIndex] == target) {

// return the index at which the value was found

return lastIndex;

}

// case: the entire array has been searched

if((theArray + lastIndex) == theArray) {

// if we are looking at the first element of the array and still . // haven’t found the target, it is not in the array

// report that target was not found in the array

return -1;

}

// try again at the next index (closer to beginning of the array)

return findByTailRecursion(theArray, (lastIndex – 1), target);

}

Tail recursion is defined on: (Page 87)