a)

Claim: Let ω be a string of symbols in the alphabet Σ. If ω is a nested string and the algorithm isNested is executed on input ω then this computation eventually halts, and the value true is returned as output.

Proof. The result will be established by induction on the length of the string ω. The strong form of mathematical induction will be used, and the cases that the length is 0, 1, and 2 will be considered in the basis.

Basis: Suppose that ω is a nested string whose length is zero. The definition of a nested string is axbyc, where x and y are open and close brackets respectively or null, and a, b, and c are all also nested strings. By following the code for isNested, we can see that it would return true, as it would do the test at line 2, where symbols.length = 0, thus i<0 is false, and the while loop breaks out. It then proceeds to line 13, and that test fails as the stack is empty since nothing has been added to it, and we get true returned at line 15 at the end of the program as is required. Since the program does not contain any brackets, x and y are null, and if x and y are null then the program will always be true as it must always be nested if there are no brackets.

Suppose that ω is a nested string whose length is 1. By following the definition of a nested string, there must be even number of brackets, one opening and one closing. Since length is equal to one, that means there will be no brackets in the string. By tracing the code we can see the test at line 2 will pass, and it will follow into the while loop. Since it is a nested string, it will not contain a (, [,), or], and thus the if statement at lines 3 and 5, and will thus skip to line 12. After incrementing i, it will then be i=1, and thus i<symbols.length fails, and we break out of the while loop. Since nothing would be added to the stack during the while loop, the stack is empty and the test at line 13 fails, and we get true returned at line 15.

Now suppose ω is a nested string of length 2. Since this is a nested string, it must have an even amount of brackets, and thus at length 2 it must either have 2 brackets or no brackets. If there are no brackets, then the tests at lines 3 and 5 will both fail, and will follow the same pattern as the past 2 cases. If there are brackets though, then there must be an opening bracket at the first position, and the test at line 3 will pass. The stack will then have a length of 1 as the bracket will be pushed onto the stack. The loop iterates, and we continue to the second character, which will be the closing bracket. The test at line 3 will fail due to not being an opening bracket, but the test at line 5 will pass. The tests at line 6 and 8 will both fail because the stack is not empty, and the brackets must match as the string is nested. The test at line 10 will then pass, and pop the top of the stack, leaving the stack empty. The loop will then break and the test at line 13 will fail, leaving the program to return true at line 15, as is required of the program.

Inductive step:

Let k be an integer such that k>=2.

These proofs heavily borrows the structure of the sample solution as seen in Solution for Problem #10 on Tutorial Exercise #8 by Dr. Wayne Eberly for the CPSC331 class of Fall 2013.

Inductive Hypothesis: let p be a word with length less than or equal to k. If p is a nested string, and the algorithm isNested is executed on it, then the program eventually halts, and true is returned as output.

Inductive claim: Let ω be a string of symbols in the language ∑ whose length is k+1. If ω is a nested string and is used as input in the algorithm isNested, then the computation eventually halts and the value true is returned as output.

Let ω be an arbitrary symbol of strings in ∑ whose length is of k+1. String ω can either be a nested string or not a nested string.

Case 1: ω is not a nested string. If ω is not a nested string, then we do not have to worry about it as it is not part of the inductive claim, which is only looking at strings which are nested.

Case 2: ω is a nested string. Since k >= 2, that means the length of the string is greater than or equal to 3. Using the definition of a nested string, axbyc, if k is of length 3, that means a or b or c is of length 1 with x and y being brackets, or a + b + c is of length 3 if x and y or null. If x and y are null, then ω is nested because there are no brackets, and the definition states that if x and y are null then the string is always nested. If x and y are not null, then that means a+b+c =1, or one of a, b, or c is of length 1. Since we have shown that a nested string of length 1 will pass in the program, this length works. The algorithm would accept the x and y pair through the algorithm, as proven in base case of length 2.

For all nested strings k+1, that means x and y are always not null if needed to be tested. This means that a+b+c = k+1-(x+y), and since x+y =2, then a+b+c=k+1-2, or k-1. This means that each nested string, a, b, and c will be smaller than the original, and being a nested string will follow the same pattern as k+1, subtracting in length over and over, until we reach k-n, where k-n = 0, 1, or 2. Since 0, 1 and 2 are all proven cases in the base cases, this string will eventually work if it is a nested string, and will be returned as true by the algorithm.

Therefore, for all strings of length k+1, if the string is a nested string, then if run through the algorithm isNested, it will return true.

b) Assume the program returns true. If the program returns true, then it must reach line 15 as that is the only place in the program in which there is a return true statement. This also means it does not return false, which means it never reaches lines 7, 9, or 14. To not reach these lines, that means it will never pass the tests of lines 6, 8, or 13. Line 6 tests if there is a closing bracket without an opening bracket, which would make the string to not be nested. Line 8 tests to check if the opening and closing brackets do not match, and if they do match it does not pass. To be nested the brackets must match each other, and not have a mismatching closing and opening pair, thus this failing this test means the string is still nested. The test at line 13 passes if there are opening brackets left while there are no closing brackets for it. To be nested there must be a closing bracket for every opening bracket, and if there is extras of

These proofs heavily borrows the structure of the sample solution as seen in Solution for Problem #10 on Tutorial Exercise #8 by Dr. Wayne Eberly for the CPSC331 class of Fall 2013.

one bracket, then it is not nested. Since it returns true, that means it passes these tests, and the string must be nested as it is not marked as not nested.

c) To prove that the algorithm isNested properly solves the 'Properly nested string recognition problem', we must show that the pre-condition and postcondition are both satisfied. The precondition states that input ω is in ∑. The postcondition states that if ω is properly nested, then true is returned as output. If ω is not properly nested, then false is returned. Since ∑ is the language of all possible inputs, there is nothing that can be input that is not in ∑, and thus the pre-condition is satisfied. Using the proof from part a, we can use it to show the first part of the post-condition, that if ω is properly nested, then true is returned. This had been proven earlier in part a using induction. The second part of the post-condition can be proved using part b. Part b proves that for true to be returned, ω must be properly nested. This also shows that if false is returned, then ω must not be properly nested as if it were properly nested, true would have been returned. Using the two former proofs, we can satisfy the post-condition. Since both the pre-condition and the post-condition are satisfied, the algorithm is solves the problem asked of us.

These proofs heavily borrows the structure of the sample solution as seen in Solution for Problem #10 on Tutorial Exercise #8 by Dr. Wayne Eberly for the CPSC331 class of Fall 2013.