# CS 211 Midterm

Notes:

* You will be asked to solve the coding challenges below during lab on 10/15.
* During that lab, you will not be allowed to use internet resources or have notes of any kind.

## Challenges

1. Write a function that finds the Nth smallest item in an array. You are **not allowed** to sort the array (using STL sort or any other sorting algorithm) beforehand.

**Example:** Given the array {1, 3, 2, 5, 9, 8, 6}, the following values would be returned given the following inputs for N:

N = 1 -> 1; N = 2 -> 2; N = 3 -> 3; N = 4 -> 5…

int findNthSmallest(vector<int> numbers, int n)

2. Given the following definition of a Binary Node:

|  |
| --- |
| class BinaryNode{  public:  int value;  BinaryNode \*left;  BinaryNode \*right;  } |

Write a **recursive** function that uses *cout* to output the BST in ***reverse sorted order*** (largest to smallest).

void reverseOrderTraversal(BinaryNode \*root)

3. Given the following definition of a Linked List

|  |
| --- |
| class LinkedList{  public:  int value;  LinkedList\* next;  }; |

Write a function that reverses the order of the linked list.

LinkedList\* reverseLinkedList(LinkedList\* root)

4. Write a function that sorts an STL stack from smallest to largest. You may use one additional stack for this operation but no other data structure (e.g. array) or sorting algorithm.

vector<int> sortStack(stack<int> &some\_stack)

The next few questions use the following class definition to build binary trees:

|  |
| --- |
| class TreeNode { public:  int value;  TreeNode \*left = nullptr;  TreeNode \*right = nullptr;  TreeNode(int v = 0)  {  value = v;  } }; |

5. Write a function called searchTree that searches a binary search tree for the supplied value. Find should return Boolean TRUE when the value is found and Boolean FALSE when it is not.

6. Write a function called bstToVector that converts and returns the supplied binary search tree into a sorted STL vector (HINT: a certain traversal makes this much easier).

7. Write a function called isAvl that returns true when the supplied tree is AVL compliant.

8. Write a function called getHeight that returns the height of the supplied tree.

9. Write a function called mergeSorted that accepts two sorted STL vectors of integers and returns a merged sorted vector. Note that you are **not allowed** to use a sorting algorithm to solve this problem.

//DONE

10. Write a function called isBalanced that determines whether or not the following braces are balanced: (); []; {}. E.g. "(ab[cd])" is balanced, "(ab" is not balanced, "(ab[cd)e]" is not balanced.