# CSC 212 Midterm Study Guide

## 1. C++ Review, Memory, Pointers

### Memory Management (Stack vs. Heap)

Stack Memory: Stores local variables, function calls, and control flow. Memory is automatically allocated and deallocated.  
Example:  
void example() {  
 int x = 10; // Stored on stack  
} // x is automatically removed from memory when function exits  
  
Heap Memory: Stores dynamically allocated data. Memory must be manually managed using new and delete.  
Example:  
int\* ptr = new int(5); // Allocated on heap  
delete ptr; // Must manually free memory

### Pointer Operations with Examples

Declaring a pointer:  
int x = 5;  
int\* p = &x; // p stores address of x  
  
Dereferencing a pointer:  
cout << \*p; // Outputs 5  
  
Pointer arithmetic:  
int arr[3] = {10, 20, 30};  
int\* ptr = arr;   
cout << \*(ptr + 1); // Outputs 20

### Common Mistakes with Pointers

Dangling Pointer – Accessing memory after deleting it  
Memory Leak – Forgetting to delete dynamically allocated memory  
Uninitialized Pointer – Using a pointer without initializing it

### Practice Problems

1. What happens if you dereference an uninitialized pointer?  
2. Write a function that returns a dynamically allocated array of n integers.  
3. What's wrong with this code?  
int\* p;  
\*p = 10;

## 2. Introduction to Analysis of Algorithms

### What is Algorithm Analysis?

Measures efficiency in terms of time complexity and space complexity. Allows for comparison of different algorithms.

### Understanding Best, Worst, and Average Case

Best-case: Minimum operations needed (e.g., an already sorted list in sorting algorithms).  
Worst-case: Maximum operations needed (e.g., reversed list in sorting algorithms).  
Average-case: Expected performance across inputs (e.g., random lists).

### Asymptotic Notation

Big-O (O(n)): Worst-case upper bound  
Big-Theta (Θ(n)): Tight bound (average case)  
Big-Omega (Ω(n)): Best-case lower bound

### Example Problems

1. What is the best, worst, and average case complexity of binary search?  
2. Prove that a loop running n times is O(n).

## 3. Big-O Notation (Extra Detailed Section)

Time and space complexity  
Common complexities explained  
How to compute Big-O for different functions  
Step-by-step examples  
C++ implementation  
Visual representations

## 4. Dynamic Arrays

Difference between static and dynamic arrays  
Resizing strategies and performance impact  
C++ implementation  
Practice problems

## 5. Stacks

What is a stack?  
Stack operations (push, pop, peek)  
Implementing stacks using arrays and linked lists  
Real-world use cases  
Practice problems

## 6. Midterm Practice Problems (All Topics)

Mixed questions covering everything  
Space for you to solve  
Answer key at the bottom

## 7. Answer Key

1. O(log n)  
2. O(n³)  
3. Use vector<int> stack; with push\_back() and pop\_back()  
4. Best-case: O(n log n), Worst-case: O(n²)