COEN 79 – Study Guide

Chapter 1

- What are the phases of software development?
- What are the two primary considerations for selecting suitable subtasks?
- Writing ‘precondition’ and ‘postcondition’ for sample functions.
- What is the meaning of “procedural abstraction”?
- Using the Standard Name Space.
- Runtime analysis of algorithms (cubic, quadratic, linear, logarithmic, etc.)
- To analyze an algorithm, why do we count the number of operations instead of measuring the actual execution time?
- Code testing methods.

Chapter 2

- What is an ‘abstract data type (ADT)’?
- What is the difference between a ‘class’ and an ‘object’?
- Class access modifiers (private and public).
- The ‘scope resolution operator’.
- What are the properties of a constructor?
- What are the pros and cons of using ‘inline member functions’?
- Using ‘namespaces’.
- How can we use namespaces to avoid name conflicts?
- How can macro-guard be used to avoid the ‘duplicate class definition problem’?
- What is the value semantics of a class?
- Present examples for: Value Parameters, Reference Parameters, and Const Reference Parameters.
- What is the meaning of ‘operator overloading’?
- How would you decide if an operator must be implemented as a member or a non-member function?
- What is a ‘friend function’?

Chapter 3

- Using ‘typedef’.
- What is the ‘size_t’ data type?
- What is a ‘const member function’?
- What is a ‘static member variable’.
- Define the “invariant” of a bag class.
- Where is the “invariant of a class” documented? Why?
- Implementation of bag class using an array (static array).
- What are the time complexities of the functions performing operations on the static bag class?
- What the “value semantics” of a class is?

Chapter 4

- Declaration and use of pointers.
- Accessing arrays via pointers.
- What is a dynamic variable?
- Difference between heap and stack memory.
- What are the causes of stack overflow?
- Why is it a good idea to use 'try-catch' when allocating dynamic memory?
- Pointers as parameters: Value Parameter that is Pointer; const Value Parameter that is Pointer; Reference Parameter that is Pointer
- Implementation of bag class using a dynamic array. Especially constructor, destructor, assignment operator, copy constructor, and reserve function.
- Why the "automatic assignment operator" fails for the dynamic bag?
- What are the different ways to activate the destructor of a class?
- What are the different ways to activate the copy constructor of a class?
- What STL classes store elements in a dynamic array?
- Implement a smart pointer for a data type.
- What are the benefits of developing a 'string' class compared to simply using arrays to represent strings? (compare std::string vs. strings in C language)
- What is the 'this' keyword, and how is it used?

Chapter 5

- Implementation of node class for building linked lists.
- Why do we need to provide two versions of functions that return a pointer or reference?
- What are the limitations of "const pointers"?
- Why linked-list toolkit functions are not member functions of the node class?
- Implement linked list toolkit functions such as: computing length, adding to the list, removing from list, copying a list, searching in a list.
- Understanding the parameter types used for implementing linked list toolkit functions.
- Loop detection in a linked list.
- Implementation of bag class with a linked list.
- Comparing the performance of bags using linked list and array.

Chapter 6

- What are the limitations of using 'typedef'?
- What are the benefits of using template functions and classes?
- What is the meaning of 'instantiation' when using template classes?
- Implement template versions of bag classes.
- Implement template versions of the 'node' class.
- What is an 'iterator'?
- Compare 'input iterator', 'output iterator', 'forward iterator', 'bidirectional iterator', and 'random access iterator'.

- Is a pointer a random-access iterator? Why?
- How are iterators used for STL's vector, forward_list, and list classes?
- What are the iterator invalidation rules for STL's vector and list?
- Implement an iterator for a singly-linked list.

Chapter 7
- Object-oriented implementation of a stack template class using array and linked list.
- Write the pseudo-code of a program to check the balance of parentheses of a mathematical expression. How many stacks are required?
- Write the pseudo-code of a program to evaluate a fully parenthesized mathematical expression. How many stacks are required?
- Write the pseudo-code of a program to convert a fully parenthesized mathematical expression to post-order notation. How many stacks are required?
- Write the pseudo-code of a program to evaluate a post-order mathematical expression. How many stacks are required?

Chapter 8
- Object-oriented implementation of a queue template class using an array (circular) and linked list.
- What are the benefits of the deque class? Compare it with bag using dynamic array and bag using linked list.
- Implementation of STL deque class. Especially the constructor, destructor, clear, pop_front, pop_back, push_front, push_back.

Chapter 9
- What is a 'full binary tree'? What is a 'complete binary tree'?
- What is the minimum required number of nodes to build a 'full binary tree' with depth d?
- What is the minimum required number of nodes to build a 'complete binary tree' with depth d?
- What are the pros and cons of using arrays to represent binary trees?
- Write non-member functions (recursive) to perform the following operations on trees: clear, copy, flip, in-order traversal, pre-order traversal, post-order traversal.
- Show how a function parameter can be a function's template parameter.
- Implement a bag class using a binary search tree. Essential functions are adding, searching, and removal.
- Write a function to find and remove the max element of a sub-tree in a binary search tree.
- What are the time complexities of adding, searching, and removing elements with a bag that stores elements in a binary search tree?

Chapter 10
- What is a 'max heap'? What is a 'min heap'?
- Is an array a suitable underlying data structure to store a heap's elements? Why?
- Addition and removal of elements with a max heap.

- What is the meaning of ‘re-hipification upward’?
- What is the meaning of ‘re-hipification downward?
- What type of data structure does the ‘STL priority queue’ implement?
- What is the ‘balance factor’ of a binary tree?
- How ‘left rotation’, ‘right rotation’, ‘left-right rotation’, and ‘right-left rotation’ are performed by the AVL tree? Write codes to implement these functions.
- How would you decide what type of rotation is necessary to fix an unbalanced binary search tree? Write a code to present your solution.
- Add to and remove elements with AVL tree.
- What are the rules of the B-tree?
- What is the ‘branching factor’ of a B-tree?
- Why MAX = 2 * MIN in B-trees?
- Write a class definition (no implementation) and present the member variables of a B-tree.
- Insert to and remove elements from B-tree.

Chapter 11

- Class access modifiers (private, protected, and public).
- How the ‘private’, ‘protected’, and ‘public’ members of a base class can be accessed by a derived class under ‘private’, ‘protected’, and ‘public’ inheritance? You must be able to draw a diagram to answer this question.
- What is the sequence of activating constructors of the base and derived classes?
- What is the sequence of activating destructors of the base and derived classes?
- What is ‘upcasting’? Present an example. Is ‘upcasting’ limited to ‘public inheritance’?
- What is the meaning of ‘overriding’ an inherited member function?
- What is the ‘member initialization list’. Present an example.
- What is a ‘virtual member function’?
- Present a ‘protected inheritance’ scenario where the derived class overrides the member functions of the base class.
- Present an example to show how the ‘virtual’ keyword is used when overriding a function.
- What is a ‘pure virtual function’?
- What is an ‘abstract class’?