CS 372 (Advanced Programming and Algorithms) Challenge Exam – Study Guide

Textbooks (available in the Regis bookstore):

Malik, D.S. (2009). *C++ Programming: From Problem Analysis to Program Design, 4th edition*. Boston, MA: Course Technology/Cengage Learning; ISBN: 1-4239-0209-2.

Gilberg, R. F. and Fourouzan, B. A. (2001). *Data Structures: A Pseudocode Approach with C++.* Brooks/Cole; ISBN: 0-534-95216-X.

Other Resources

Additional topics are covered in the **CS372 Supplemental Materials**, which can be downloaded from: http://www.regis.edu/regis.asp?sctn=cpcis&p1=ap&p2=cs&p3=mod

CBE Exam

The CS372 exam consists of 2 parts.

The first part is a written exam. You will have 3 hours to complete the written exam.

NOTE: Students **may** use a calculator during the exam.

The second part is a take-home programming exam. You will have 3 weeks to complete the programs. Your programs should be coded using an ANSI C++ compiler (Dev-C++ is recommended) in CONSOLE mode (WITHOUT using any object-oriented features).

The Dev-C++ compiler may be downloaded free from:

http://www.bloodshed.net/dev/devcpp.html

You must design and write the program code and return it to the testing office in the allotted time. You may be required to turn in any or all of the following as part of the take-home exam:

1) Source code (both a hardcopy, and on diskette or CD)
2) Executable code
3) Analysis report
4) Structure chart
5) Test Plan

CS 372 (Advanced Programming and Algorithms) Challenge Exam – Study Guide

The CS372 exam may cover any of the following Programming and Algorithm topics:

- Algorithm Analysis and big-O notation
- Abstract Data Types (ADTs)
- Linked lists including:
    - Doubly linked lists
    - Circular lists
- Recursion
- Searching Algorithms
- Hashing Methods
- Hashing Collision Resolution including:
    - Linear Probing
    - Quadratic Probing
    - Chained Lists
- Sorting algorithms including:
    - Selection sort
    - Bubble sort
    - Insertion sort
    - Shell sort
    - Merge sort
    - Quick sort
- Stack Applications including:
    - Reversing Data
    - Parsing Data
    - Postponement (e.g. Infix to Postfix conversion)
    - Backtracking
- Queue Applications including:
    - Data Categorization
    - Queuing Simulations
    - Controlling Tasks
- Design, implementation, testing, and debugging of programs
- Program testing and analysis

CS 372 (Advanced Programming and Algorithms) Challenge Exam – Study Guide

Details on Course Coverage

Learning Topic #1: C++ Concepts

Upon completion of this learning topic, students will be able to:

- Compare and contrast C strings and the C++ string type.
- Explain the data structures utilized in command line arguments.
- Develop software that passes data into programs using command line arguments.
- Compile and run programs contained in multiple files.

Learning Topic #2: ADTs and Linked Lists

Upon completion of this learning topic, students will be able to:

- Explain the concepts of the Abstract Data Type (ADT) and how it applies to a linked list.
- Describe the linked list data structure.
- Detail typical operations on a linked list.
- Compare and contrast more complex implementation of the linked list data structure (circular linked list, doubly-linked list, multi-linked list).
- Develop software to create and access data within a dynamic memory allocation implementation of a linked list data structure.

Learning Topic #3: Algorithm Efficiency and Big-O Notation

When there are two different algorithms that solve the same problem, one algorithm often performs significantly better than another. As a Computer Scientist, it is vital that students have the ability to analyze a piece of code or an algorithm and understand its efficiency, so that they can recognize and choose the more efficient algorithm.

An algorithm analysis should focus on **significant** differences that are likely to dominate the overall efficiency. Otherwise, one could select an algorithm that runs a fraction of a

second faster than another algorithm, yet requires many more hours of your time to implement and maintain.

Although the efficiency equation derived for a function can be complex, we can examine the **dominant factor** in the equation to determine the relative **magnitude** of the efficiency. Hence, we don't need to determine the complete measure of efficiency, only the factor that determines the magnitude. This dominating factor is the **big-O** and is expressed as **O(n).** Big-O means "**on the order of**".

This topic will cover how to analyze an algorithm to determine its efficiency, using big-O notation.

Upon completion of this learning topic, students will be able to:

- Understand why it's important to study an algorithm's efficiency.
- Analyze an algorithm to determine its efficiency and explain how Big-O notation is used to illustrate algorithm efficiency.
- Be able to analyze different algorithms and determine the most efficient among them.

Learning Topic #4: Large Project Design and Implementation

Upon completion of this learning topic, students will be able to:

- Detail the necessary steps required in Large Software Project Design and Implementation.
- Develop code as part of a pair/team.

CS 372 (Advanced Programming and Algorithms) Challenge Exam – Study Guide

Learning Topic #5: Basic Searching Algorithms

Searching is one of the most used tasks in software programs. In almost every software program there is a need to either search to find, search not to find, search to change, search to add, or search to delete data from a list.

This topic will cover the following searching algorithms:

- Sequential Search
- Sentinel Search
- Probability Search
- Ordered List Sequential Search
- Binary Search

Upon completion of this learning topic, students will be able to:

- Find a particular item within a list data structure using various searching methods.
- Analyze and compare searching methods to determine algorithm efficiency.

Learning Topic #6: Searches with Hashing

Upon completion of this learning topic, students will be able to:

- Compare and contrast the different Hashing methods.
- Arrange item keys into a hash table list using various Hashing and Collision Resolution methods.
- Analyze and compare searching methods to determine algorithm efficiency.

Learning Topic #7: Recursive Algorithms

Upon completion of this learning topic, students will be able to:

- Understand how recursive algorithms work and how to develop them.
- Determine when it is appropriate to use a recursive algorithm.

- Hand trace a recursive function call and be able to predict the results.
- Write a C++ program using recursive functions.
- Analyze the efficiency of recursive algorithms.

Learning Topic #8: Advanced Sorting Algorithms, Part 1

Sorting algorithms can be divided into two groups, by the complexity of their algorithms. The first group of sorting algorithms have efficiencies of **$O(n^2)$**. This group includes the Bubble sort, Insertion sort, Selection sort, and Shell sort.

Upon completion of this learning topic, students will be able to:

- Write code that will sort unordered data into a sorted list using the Selection Sort, the Bubble Sort, the Insertion Sort, and the Shell Sort.
- Analyze and compare the sorting methods to determine algorithm efficiency.

Learning Topic #9: Advanced Sorting Algorithms, Part 2

The second group of sorting algorithms performs in **O(n log n)** time, and includes the Heap sort, Merge sort, and Quick sort. Two of these **O(n log n)** algorithms, Merge sort and Quick sort, will be covered by this topic.

Upon completion of this learning topic, students will be able to:

- Write programs that will sort unordered data into a sorted list using the Merge Sort.
- Write programs that will sort unordered data into a sorted list using the Quick Sort.
- Analyze and compare various sorting methods to determine algorithm efficiency.
- Determine which sorting algorithm would be best to use for a specific set of data.
- Develop programs that use pointers to functions.

CS 372 (Advanced Programming and Algorithms) Challenge Exam – Study Guide

Learning Topic #10: Stacks and Queues

Upon completion of this learning topic, students will be able to:

- Describe the stack linear list structure.
- Detail the typical operations on a stack.
- Describe the queue linear list structure.
- Detail the typical operations on a queue.
- Compare and contrast the implementation of the stack and the queue as either a sequential list or a linked list.
- Develop software to create and access either a stack or a queue.