BUTTE COLLEGE
COURSE OUTLINE

I. **CATALOG DESCRIPTION**

CSCI 21 - Programming and Algorithms II **3 Unit(s)**

Prerequisite(s): CSCI 20
Recommended Prep: NONE
Transfer Status: CSU/UC
34 hours Lecture
51 hours Lab

This is a software engineering course, focused on the application of software engineering techniques for the design and development of large programs. Topics include data abstraction, data structures and associated algorithms, recursion, declaration models, and garbage collection. Students will learn to design, implement, test, and debug programs using an object-oriented language. (C-ID COMP 132).

II. **OBJECTIVES**

Upon successful completion of this course, the student will be able to:

A. Design and implement programs that use arrays, records/structs, strings, linked lists, stacks, queues, hash tables, and trees.
B. Design, implement, test, and debug recursive functions and procedures.
C. Evaluate the tradeoffs in lifetime management of data when using manual memory management versus reference counting or tracing garbage collection.
D. Explain how abstraction mechanisms support the creation of reusable software components.
E. Design, implement, test, and debug programs in an object-oriented language.
F. Compare and contrast object-oriented analysis and design with structured analysis and design.

III. **COURSE CONTENT**

A. **Unit Titles/Suggested Time Schedule**

Lecture

Topics	Hours
1. Primitive types, arrays, and records/structs	2.00
2. Strings and string processing	2.00
3. Memory: data representation, static/stack/heap allocation, runtime management	1.00
4. Pointers and references	1.00
5. Linked structures	4.00
6. Stacks, queues, hash tables	4.00
7. Trees	2.00
8. Selecting data structures	1.00
9. Recursion: mathematical functions and simple procedures	1.00
10. Divide-and-conquer strategies, backtracking	1.00
11. Implementing recursion	1.00
12. Variable binding, visibility, scope, lifetime, and type-checking	1.00
13. Garbage collection	0.50
14. Procedures, functions, and iterators	1.00
15. Parameterization mechanisms	0.50

16. Activation records and storage management	1.00
17. Type parameters and parameterized types: templates and generics	2.00
18. Object-oriented design	2.00
19. Encapsulation, information hiding, and separation of behavior and implementation	1.00
20. Classes, subclasses, inheritance, class hierarchies	2.00
21. Polymorphism	2.00
22. Collection classes and iteration protocols	1.00
Total Hours	34.00

Lab

Topics	Hours
1. Primitive types, arrays, and records/structs	3.00
2. Strings and string processing	3.00
3. Pointers and references	1.50
4. Linked structures	6.00
5. Stacks, queues, hash tables	6.00
6. Trees	6.00
7. Implementing recursion	3.00
8. Variable binding, visibility, scope, lifetime, and type-checking	1.50
9. Procedures, functions, and iterators	1.50
10. Parameterization mechanisms	1.50
11. Type parameters and parameterized types: templates and generics	6.00
12. Object-oriented design	3.00
13. Classes, subclasses, inheritance, class hierarchies	6.00
14. Polymorphism	3.00
Total Hours	51.00

IV. **METHODS OF INSTRUCTION**

A. Lecture
B. Collaborative Group Work
C. Homework: Students are required to complete two hours of outside-of-class homework for each hour of lecture
D. Demonstrations
E. Multimedia Presentations

V. **METHODS OF EVALUATION**

A. Quizzes
B. Homework
C. Lab Projects
D. Mid-term and final examinations

VI. **EXAMPLES OF ASSIGNMENTS**

A. Reading Assignments
1. Read the section in your text on linked list nodes. Write the code to define a struct for a

node for a singly-linked list, then test the struct in a driver by creating a static and a dynamic instance of the struct.
    2. Read the Application Programmer's Interface (API) for the C++ Standard Template Library (STL) stack and queue classes at cplusplus.com. List and briefly describe all of the member functions that these classes have in common, and describe how the push and pop functions work for each.

B. Writing Assignments
1. Write the pseudocode for two versions of an algorithm to delete all of the nodes from a doubly-linked list. One version must use an iterative approach, while the other must use a recursive approach.
2. Write complete documentation for the sample header file provided by the instructor, using the associated implementation file as a reference. Be certain to document return values and parameters of each function, as appropraite, and provide a thorough description of the purpose of each function. Your documentation must adhere to the style guidelines for the class.

C. Out-of-Class Assignments
1. Use the Internet to research reference counting and tracing as garbage collection techniques. Write a short summary of your findings, including a list of five programming languages (of your choice) and a description of the garbage collection technique(s) supported by each.
2. Create a Unified Modeling Language (UML) diagram for a simple inventory management system for a coffee stand that has three types of coffee, four types of snacks, and cups/napkins/spoons. All of the items will be derived from a single base class, InventoryItem (price, quantity, supplier), all of the coffee will be derived from a single Coffee class, all of the snacks will be derived from a single Snack class, and cups/napkins/spoons will be derived from a single Supply class. Give every class in your hierarchy 1-2 unique attributes.

VII. RECOMMENDED MATERIALS OF INSTRUCTION

Textbooks:

A. Savitch, Walter. Absolute C++. 6th Edition. Pearson, 2015.

B. Carrano, Frank M. Data Abstraction and Problem Solving with C++: Walls and Mirrors. 7th Edition. Pearson, 2016.

Created/Revised by: John Trolinger
Date: 03/07/2016