CS 10B Course Outline as of Fall 2018

CATALOG INFORMATION

Dept and Nbr: CS 10B Title: PROGRAMMING CONCEPTS 1
Full Title: Programming Concepts and Methodologies 1
Last Reviewed: 2/8/2021

Units		Course Hours per Week		Nbr of Weeks	Course Hours Total	
Maximum	4.00	Lecture Scheduled	3.00	17.5	Lecture Scheduled	52.50
Minimum	4.00	Lab Scheduled	3.00	6	Lab Scheduled	52.50
		Contact DHR	0		Contact DHR	0
		Contact Total	6.00		Contact Total	105.00
		Non-contact DHR	0		Non-contact DHR	0

Total Out of Class Hours: 105.00 Total Student Learning Hours: 210.00

Title 5 Category:	AA Degree Applicable
Grading:	Grade or P/NP
Repeatability:	00 - Two Repeats if Grade was D, F, NC, or NP
Also Listed As:	
Formerly:	

Catalog Description:
Introduces the discipline of computer science using C++ and utilizing programming and practical hands-on problem solving.

Prerequisites/Corequisites:
Course Completion of CS 10A

Recommended Preparation:
Eligibility for ENGL 1A or equivalent

Limits on Enrollment:

Schedule of Classes Information:
Description: Introduces the discipline of computer science using C++ and utilizing programming and practical hands-on problem solving. (Grade or P/NP)
Prerequisites/Corequisites: Course Completion of CS 10A
Recommended: Eligibility for ENGL 1A or equivalent
Limits on Enrollment:
Transfer Credit: CSU;UC.
Repeatability: Two Repeats if Grade was D, F, NC, or NP

ARTICULATION, MAJOR, and CERTIFICATION INFORMATION:

AS Degree:	**Area**	Effective:	Inactive:
CSU GE:	**Transfer Area**	Effective:	Inactive:
IGETC:	**Transfer Area**	Effective:	Inactive:

CSU Transfer:	Transferable	Effective:	Fall 2018	Inactive:	
UC Transfer:	Transferable	Effective:	Fall 2018	Inactive:	

CID:
CID Descriptor: COMP 122 Programming Concepts and Methodology I
SRJC Equivalent Course(s): CS10A OR CS10B

Certificate/Major Applicable:
Both Certificate and Major Applicable

COURSE CONTENT

Student Learning Outcomes:
At the conclusion of this course, the student should be able to:
1. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.
2. Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems.
3. Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.
4. Demonstrate different forms of binding, visibility, scoping, and lifetime management

Objectives:
Upon completion of this course students will be able to:
1. Choose appropriate conditional and iteration constructs for a given programming task.
2. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
3. Identify the necessary properties of good algorithms.
4. Create algorithms for solving simple problems.
5. Identify at least one distinguishing characteristic for each of the programming paradigms covered in this unit.
6. Explain the value of declaration models, especially with respect to programming-in-the-large.
7. Identify and describe the properties of a variable such as its associated address, value, scope, persistence, and size.
8. Describe strategies that are useful in debugging.

Topics and Scope:

I. Fundamental Programming Constructs
   A. Basic syntax and semantics of a higher-level language

B. Variables, types, expressions, and assignment
C. Simple I/O
D. Conditional and iterative control structures
E. Functions and parameter passing
F. Structured decomposition

II. Algorithms and Problem-Solving
A. Problem-solving strategies
B. The role of algorithms in the problem-solving process
C. Implementation strategies for algorithms
D. Debugging strategies
E. The concept and properties of algorithms

III. Overview of Programming Languages
A. History of programming languages
B. Brief survey of programming paradigms
C. Procedural languages
D. Object-oriented languages

IV. Declarations and Types
A. The conception of types as a set of values together with a set of operations Declaration models (binding, visibility, scope, and lifetime)
B. Overview of type-checking

All topics are covered in both the lecture and lab parts of the course.

Assignment:

Lecture Related Assignments:
1. Read approximately 30 pages per week
2. Complete 2-8 examinations including final exam

Lab Related Assignments:
1. Complete 10-15 programming assignments,with documentation, using the C++ programming language

Methods of Evaluation/Basis of Grade:

Writing: Assessment tools that demonstrate writing skills and/or require students to select, organize and explain ideas in writing.

Written program documentation	Writing 10 - 20%

Problem Solving: Assessment tools, other than exams, that demonstrate competence in computational or non-computational problem solving skills.

Programming assignments	Problem solving 20 - 60%

Skill Demonstrations: All skill-based and physical demonstrations used for assessment purposes including skill performance exams.

None	Skill Demonstrations 0 - 0%

Exams: All forms of formal testing, other than skill performance exams.

Exams, Final Exam: (Multiple choice, true/false, matching items, completion, programming problems)	Exams 20 - 60%

Other: Includes any assessment tools that do not logically fit into the above categories.

None	Other Category 0 - 0%

Representative Textbooks and Materials:
Starting Out with C++ From Control Structures through Objects. 8th ed. Gaddis, Tony. Pearson. 2014