COURSE NUMBER/TITLE: CS-145 [354-145] COMPUTER SCIENCE II

CREDITS: 4

COURSE DESCRIPTION: Basic concepts in computer systems including computer structure, assembly languages, addressing techniques, program segmentation and linkage. Problem solving using a high-level programming language. Procedural programming concepts, string processing, searching and sorting algorithms, pointers, dynamic memory allocation, memory management, linear data structures, recursive structures, object-oriented programming, and debugging. Prerequisite: CS-144 Computer Science I

TEXTBOOK: The C Programming Language, 2nd Ed., by Kernighan & Ritchie (adopted Spring 14) 
Starting Out with C++: Early Objects, 8th Ed., by Gaddis (adopted Spring 14) 
Assembly Language for x86 Processors, 6th Ed., by Irvine (adopted Spring 12) 

Previous: 
Starting Out with Java, 2nd Ed., by Gaddis (adopted F11) 
JAVA-How to Program, 6th Ed., by Dietel (adopted S05) 
JAVA: An Introduction to, 2nd Ed., by Savitch (adopted S02) 
The Object Concept, 1st Ed., by Decker (Prior to F96) 
Pascal Programming and Problem Solving, 4th Ed., by Leetsma (Prior to F95) 
Advanced Programming in Pascal with Data Structures, 1st Ed., by Nyhoff (Prior to F95)

COURSE OBJECTIVES:
Upon successful completion of the course, the student will be able to:
1. Demonstrate an understanding of the basic structure of computer systems at the conventional machine level and familiarity with assembly language programming of a modern computer system.
2. Display knowledge of techniques for algorithm development and problem solving, and write their own complex solutions.
3. Demonstrate a deep understanding of programming design and implementation, and debugging and testing tools.
4. Demonstrate an increased level of sophistication in communication skills, including being able to document programs, listen to and speak about technical materials, and read texts to learn information independently.

COURSE OUTLINE:
1. Assembly language (Objective 1)
   a. Assembly language fundamentals: basic elements, assembling, linking, and running programs
   b. Conditional processing: Boolean and comparison instructions
   c. Integer arithmetic (Bit operations): shift, rotate, multiplication and division instructions
   d. High-level language interface: inline assembly codes and linking to high-level languages like C/C++
2. Algorithm development and problem solving (Objective 2)
   a. Sorting and searching algorithms
b. High-level language expression evaluating algorithms
c. Generic algorithms to solve complex problems

3. Programming concepts (Objectives 1, 3, 4)
   a. Functions and passing arguments
   b. Pointers
   c. String processing
d. File handling
e. Friend functions
f. Operator overloading
g. Arrays and vectors
h. Bitwise operators and bit manipulation techniques
i. Memory management
j. Dynamic memory allocation
k. Linked lists
l. Containers and collections
m. Recursive structures
n. Exception handling
   o. Conversion rules in high-level languages like C/C++

4. Advanced topics in objective-oriented programming (Objectives 3, 4)
   a. Inheritance, interfaces, polymorphism
   b. Aggregation
   c. Class templates in high-level languages like C++

5. Testing tools to speed up debugging and error analysis (Objectives 3, 4)