Course Objective and Outcome Form

Department of Electrical and Computer Engineering

North South University, Bashundhara, Dhaka-1229, Bangladesh

Course Number and Title: CSE 115 Programming Language I

Credits: 4SCH (3SCH Lecture + 1 SCH Lab)

Course type: Required, Engineering, Lecture

Course Prerequisites: None

Course Schedule/Timing: Lecture – 3 Hours/Week (Lecture), 3 Hours/Week (Lab)

Instructor(s)-in-charge: Dr. Ahsanur Rahman

Course Assessment:

Lecture:
Exam: Quizzes – 5 (two are mandatory), Midterm – 1, Final - 1

Lab:
Lab works, lab-homeworks
Midterm – 1, Final – 1, Project – 1

Grading policy:

Lecture:
Attendance - 10%, Quiz – 25%,
Midterm1-30%, Final – 35%

Lab:
Lab-practice & try-yourself – 8%, Lab quiz - 32%,
Lab Midterm-20%, Lab Final exam– 25%,
Project – 10%, Lab assignments– 5%

Course Objective

Upon completion of the course, the student should be able to:

1) Learn fundamental knowledge on basics of computers hardware and number systems.
2) Familiarize about the basic terminologies used in computer programming.
3) Understand, analyze and implement software development tools like algorithm, pseudo codes and programming structure.
4) Proficiently transform designs of problem solutions into a standard programming language.
5) Use an integrated programming environment to write, compile, and execute programs involving a small number of source files.
6) Proficiently use fundamental programming elements including: variable declaration, use of data types and simple data structures (arrays, strings and records), decision structures, loop structures, functions/methods, input and output for console and text files.

7) Apply debugging and testing techniques to locate and resolve errors, and to determine the effectiveness of a program.
8) Have understanding of professionalism, codes of ethics and responsible conduct.

Catalog Description(Syllabus**):** This is the first course in the computer science programming sequence and is required of all computer science and engineering major with no prior programming experience. This course introduces the fundamental concepts of structured programming. Topics include fundamental programming constructs: syntax and semantics of a higher-level language, variables, expressions, and assignment, simple I/O to console and files, conditional and iterative control structures, functions and parameter passing, dynamic memory allocation; Fundamental data structures: arrays, records, strings and string processing; Software development methodology: Fundamental design concepts and principles, testing and debugging strategies; Fundamental of computers; Professionalism, codes of ethics and responsible conduct, copyrights, intellectual property, and software piracy.

Textbook and related course materials:

- **"Problem Solving and Program Design in C", 7th edition by J Hanly and E Koffman, Pearson**
- **Schaum's outlines "Programming with C", 2nd edition by Byron Gottfried**
- "C: The Complete Reference", 4th edition by Herbert Schildt, Osborne/McGraw-Hill
- "Let us C", 8th edition, Y. Kanetkar, 2005
- "C: How to Program", 7th edition by P Deitel and H Deitel, Prentice Hall

Topics covered and level of coverage (Topic/Hours**):**

Course Topics	**Coverage**
Number Systems, computer hardware and software, software development method and professional ethics for computer programmers.	4.5 Hours(Lecture)
C language elements, variable declarations and data types, executable statements, general form of a c program, arithmetic expressions, formatting numbers in program output, basic input/output and common programming errors. **Keywords:**int, long, char, float, double, signed, unsigned, printf, scanf **Operators:**arithmetic(+, -, *, /, %), assignment (=), others (+=, -=, *=, /=, %=)	3 Hours (Lecture) 4.5 Hours (Lab)
Top-down design with functions:Building programs from existing information, library functions, top-down design and structure charts, functions without arguments and functions with input arguments. **Keywords:**void, return **Others:** Flow chart, pseudo code	3 Hours (Lecture) 3 Hours (Lab)

Selection structures: Control structures, conditions, the if statement, if statements with compound statements, decision steps in algorithms, nested if statements and multiple-alternative decisions and the switch statement. **Keywords:**if, if else, if else if else, switch, case, break, default **Operators:**logical(!,&&, ||), relational (<, <=, >, >=, ==, !=), others (?:)	4.5 Hours (Lecture) 4.5 Hours (Lab)
Repetition and loop statements: repetition in programs, counting loops and the while statement, computing a sum or a product in a loop, the for statement, conditional loops, loop design, nested loops, the do-while statement and flag-controlled loops. **Keywords:**for, while, do while, continue, goto **Operators:**increment and decrement (++, --)	3 Hours (Lecture) 4.5 Hours (Lab)
Pointers, modular programming: Pointers and the indirection operator, functions with output parameters, multiple calls to a function with input/output parameters, scope of names, and formal output parameters as actual arguments. **Operators:**address(&), pointer (*), dereference (*)	3 Hours (Lecture) 3 Hours (Lab)
Arrays and Strings: Declaring and referencing arrays, array subscripts, using for loops for sequential access, using array elements as function arguments, array arguments, searching and sorting an array, parallel arrays and enumerated types, multidimensional arrays, string basics, string library functions: assignment and substrings, string concatenation, string comparison, arrays of pointers, character operations, string-to-number and number-to-string conversions.	4.5 Hours (Lecture) 6 Hours (Lab)
Recursion: Nature of recursion, tracing a recursive function, recursive mathematical functions, recursive functions with array and string parameters.	3 Hours (Lecture) 3 Hours (Lab)
Structure and Union Types: User-defined structure types, structure type data as input and output parameters and functions whose result values are structured. **Keywords:**struct, union, typedef, sizeof **Operators:**member selection (.,->)	1.5 Hours (Lecture) 1.5 Hours (Lab)
Text and binary file Processing: Text and binary files, input/output using files and add, delete and search in a file.	3 Hours (Lecture) 3 Hours (Lab)
Abstraction to manage complexity, personal libraries: header files, personal libraries: implementation files, storage classes, modifying functions for inclusion in a library, conditional compilation, arguments to function main, preprocessor anddefining macros with parameters.	3 Hours (Lecture) 3 Hours (Lab)

Keywords:auto, static, register, extern, const, enum **Operators:**bitwise (~, &, |, ^, <<, >>), others (&=, |=, ^=, <<=, >>=)	

Material available to students and department at the end of the course:

Course Objectives and Outcomes Form: Student, Department, Instructor
Lecture notes, homework assignments and solutions: Student, Department, Instructor
Student work sample solutions (homework, quiz, exam, report etc.): Department
Course performance form including student surveys: Department, Instructor

Involve computer assignments? Yes

Will this course have TA(s)/Lab Instructor when it is offered?No

Level of contribution of course to Learning Outcomes (a-l:Strong, average, low**)**

Learning Outcome	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
Contribution of Course	***	**			*		*		**	***	*	*

***** Strong ** Moderate *Low**

How Learning outcomes are covered by the specific course objective/outcome

Course Objective/outcome	**Learning outcome**
1) Learn fundamental knowledge on basics of computers hardware and number systems.	A, L
2) Familiarize about the basic terminologies used in computer programming.	L
3) Understand, analyze and implement software development tools like algorithm, pseudo codes and programming structure.	A, B, G, I, J, K
4) Proficiently transform designs of problem solutions into a standard programming language.	A, B, I, J
5) Use an integrated programming environment to write, compile, and execute programs involving a small number of source files.	I
6) Proficiently use fundamental programming elements including: variable declaration, use of data types and simple data structures (arrays, strings and records), decision structures, loop structures, functions/methods, input	A, B, J

and output for console and text files.	
7) Apply debugging and testing techniques to locate and resolve errors, and to determine the effectiveness of a program.	A, J
8) Have understanding of professionalism, codes of ethics and responsible conduct.	E

Student Outcome: BS -CSE Program

The BS-CSE has twelve (a-l) student learning outcomes adopted from the preferred outcomes of both Computing Accreditation Commission (CAC) and Engineering Accreditation Commission (ECA) of ABET. The students who complete the BS-CSE program will have:

(A) an ability to apply knowledge of computing, mathematics, science and engineering appropriate to the discipline

(B) an ability to analyze a problem, and identify and define the computing requirements appropriate to its solution

(C) an ability to design a computer-based system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability

(D) an ability to function effectively on multidisciplinary teams to accomplish a common goal

(E) an understanding of professional ethical,legal, security and social issues and responsibilities

(F) an ability to communicate effectively with a range of audiences

(G) the broad education necessary to analyze and understand the impact of local and global engineering and computing solutions on individuals, organizations, global, economic, environmental, and societal context

(H) a recognition of the need for, and an ability to engage in life-long learning and continuing professional development

(I) an ability to use current techniques, skills, and tools necessary for engineering and computing practice.

(J) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the trade-offs involved in design choices.

(K) an ability to apply design and development principles in the construction of software systems of varying complexity.

(L) a knowledge of contemporary issues