MULTIMEDIA UNIVERSITY®

COURSE INFORMATION

1.	**Name of Course**	Programming Language Translation
2.	**Course Code**	TCP2451
3.	**Type of Course** (e.g. : Core, major, elective etc.)	Elective
4.	**Synopsis**	This course presents the principles needed to design and implement programming language translator. The course also provides hands-on experience with compiler construction tools and techniques.
5.	**Version** (State the date of theSenate's approval - previous and the current approval date)	Current: January 2018 Previous: June 2016
6.	**Name(s) of Academic Staff**	Yeoh Eng Thiam, Nathar Shah bin Packier Muhammad
7.	**Semester and Year Offered**	Trimester 1 (Delta Level)
8.	**Credit Value**	4
9.	**Pre-Requisite**	TCP1201 Object-Oriented Programming and Data Structures

10. **Objective of the course in the programme:**

To provide a thorough introduction to the theory and practice of programming language translation and to provide extensive hands-on experience with compiler construction tools and techniques.

11. **Justification for including the course in the programme:**

This subject trains the student on the application of theoretical concepts and also provides a hands-on development of a complicated software application. The student will learn to develop a software by systematically separating it into individual components and also learn techniques that are applicable in developing other types of software.

12. **Course Learning Outcomes (CLO)**

		Domain	Level
CLO1:	Explain the process of translating a program from one language to another language	Cognitive	2
CLO2:	Apply lexical and syntax analysis techniques on a given context free grammar	Cognitive	3
CLO3:	Design lexical and syntax analyser phases of a compiler	Cognitive	6
CLO4:	Explain the processes of semantic checking, code generation and optimization in a compiler	Cognitive	2

13. **Mapping of the Course Learning Outcomes to the Programme Learning Outcomes, Teaching Methods and Assessment:**

Course Learning Outcomes (CLO) (Must tally with CLOs in item 12)	Programme Learning Outcomes (PLO)												Teaching Methods	Assessment Method
	PLO1	PLO2	PLO3	PLO4	PLO5	PLO6	PLO7	PLO8	PLO9	PLO10	PLO11	PLO12		
CLO1							✓						Lecture/Practical	Quiz/Lab/Test/Final Exam
CLO2								✓					Lecture/Practical	Assignment/Quiz/Lab/Test/Final Exam
CLO3								✓					Lecture/Practical	Assignment/Lab/Final Exam
CLO4							✓						Lecture/Practical	Quiz/Lab/Final Exam
Total							**2**	**2**					*Indicate the relevancy between the CLO and PLO by ticking "✓" the appropriate relevant box (This description must be read together with standards 2.1.2, 2.2.1, and 2.2.2 in Area 2 – pages 16 & 18 of COPPA 2.0)*	

14. **Transferable Skills:**

Analytical thinking through written work assessed by quiz, test and final exam.
Problem solving skills through practical work assessed by lab and assignment.

15. **Distribution of Student Learning Time (SLT)**

	Course Content Outline	**CLO	Teaching and Learning Activities – Guided Learning (F2F)* *L	*T	*P	*O	Guided Learning (NF2F)*	Independent Learning (NF2F)*	Total SLT
1	**Introduction to Language Translation** Language translation systems:interpreters, compilers and assemblers; Language translation phases; Machine-dependent and machine-independent aspects of translation; Main components in compilers; Review of programming language concepts.	**1**	2		2		0	4	8
2	**Lexical Analysis** Role of lexical analyzer; Specification and recognition of tokens; Application of regular expressions in lexical scanners; Implementation of finite-state automata; Handling lexical errors.	**2**	4		6		4	10	24
3	**Syntax Analysis** Role of parsers; Formal definition of context free grammars; Parse tree vs. abstract syntax tree; Recursive-descent parsing; Shift-reduce parsing; Construction of parsing tables; Eliminating ambiguity, left recursion and left factoring; Precedence and Associativity; Syntax-directed translation; Handling parsing errors.	**2**	4		8		8	12	32

4	**Semantic Analysis** Role of semantic analyzer; Symbol table management; Declaration models: binding, visibility, scope, and lifetime; Inherited and synthesized attributes; Annotated syntax tree and dependency graphs; Bottom-up and Top-down evaluation of Attributes; Data type as set of values with set of operations; Semantic models of user-defined types; Equivalence of types; Type-checking models; Type conversion; Type checking algorithms.	**4**	4		4		0	8	16
5	**Code Generation** Role of code generator; Intermediate code generation; Intermediate representations; Issues in the design of code generator; Instruction selection and register allocation; Addresses in target code; Basic blocks and flow graphs; Run-time environment; Generating codes for arithmetic expressions, Boolean expressions, control structures, and procedure calls.	**4**	2		2		0	4	8
6	**Optimization** Local and global optimization; Peep Hole optimization; Optimization of basic blocks; Optimizing using Directed Acyclic Graphs; Global redundancy and data flow analysis; Looping improvement.	**4**	2		2		0	4	8
7	**Tools for language translation** Automated generation of lexical and syntax analyzers; Specifying regular expressions for scanner generation; Specifying grammar for parser generation; Handling ambiguities and conflicts; Combining tools for an integrated compiler. Overview of automation tools: Lex, YACC, JLex, Java CUP.	**3**	4		4		0	8	16
								Total SLT	**112**

SUMMATIVE ASSESSMENT

1. Continuous Assessment	**Percentage %**	**Total SLT**
Assignment	20%	12
Quiz	5%	6
Lab	10%	6
Test	15%	4
Total SLT for Continuous Assessment		**28**

2. Final Assessment	**Percentage %**	**Total SLT**	
		F2F	**ILT**
Final Exam	50%	2	18
Total SLT for Final Assessment (F2F + NF2F)		**20**	

Grand Total	**100%**	**160**

****Indicate the CLO based on the CLO's numbering in Item 12.**
***L= Lecture, *T= Tutorial, *P= Practical, *O= Others, F2F*= Face to Face, NF2F*= Non Face to Face**

16 . **Identify Special Requirement to Deliver the Course (e.g., software, nursery, computer lab, simulation room):**
Java

17 . **Main References:**
Aho, Lam, Sethi, Ullman, Compilers: Principles, Techniques, & Tools, 2nd Ed., Addison-Wesley, 2007.

18 . **Additional References:**
Michael L. Scott, Programming Language Pragmatics, 3rd Ed., Morgan Kaufman, 2009.
Keith D. Cooper, Linda Torczon, Engineering a Compiler, 2nd Ed., Morgan Kaufman, 2012.
D. Grune, K. Reeuwijk, H. E. Bal, C. J. H. Jacobs, K. Langendoen, Modern Compiler Design, 2nd Ed., Springer, 2012.

Note:
Cells shaded light grey contain formulas / fixed values. Edit these formulas only if needed.