Course Code	18CSC207J	Course Name	ADVANCED PROGRAMMING PRACTICE	Course Category	C	Professional Core	L	T	P	C
							3	0	2	4

Pre-requisite Courses	18CSC202J	Co-requisite Courses	18CSC204J	Progressive Courses	Nil
Course Offering Department	Computer Science and Engineering	Data Book / Codes/Standards	Nil		

Course Learning Rationale (CLR):	The purpose of learning this course is to:	Learning		
		1	2	3
		Level of Thinking (Bloom)	Expected Proficiency (%)	Expected Attainment (%)
CLR-1 :	Create Real-time Application Programs using structured, procedural and object oriented programming paradigms			
CLR-2 :	Create Real-time Application Programs using event driven, declarative and imperative programming paradigms			
CLR-3 :	Create Real-time Application Programs using parallel, concurrent and functional programming paradigms			
CLR-4 :	Create Real-time Application Programs using logic, dependent type and network programming paradigms			
CLR-5 :	Create Real-time Application Programs using symbolic, automata based and graphical user interface program paradigm			
CLR-6 :	Create Real-time Application Programs using different programming paradigms using python language			

Course Learning Outcomes (CLO):	At the end of this course, learners will be able to:	Learning			Program Learning Outcomes (PLO)															
		1	2	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		Level of Thinking (Bloom)	Expected Proficiency (%)	Expected Attainment (%)	Engineering Knowledge	Problem Analysis	Design & Development	Analysis, Design, Research	Modern Tool Usage	Society & Culture	Environment & Sustainability	Ethics	Individual & Team Work	Communication	Project Mgt. & Finance	Life Long Learning	PSO - 1	PSO - 2	PSO – 3
CLO-1 :	Create Programs using structured, procedural and object oriented programming paradigms	3	85	80	H	H	H	H	H	-	-	L	M	M	L	M	-	M	-
CLO-2 :	Create Programs using event driven, declarative and imperative programming paradigms	3	85	80	H	H	H	H	H	-	-	L	M	M	L	M	-	-	-
CLO-3 :	Create Programs using parallel, concurrent and functional programming paradigms	3	85	80	H	H	H	H	H	-	-	L	M	M	L	M	-	-	-
CLO-4 :	Create Programs using logic, dependent type and network programming paradigms	3	85	80	H	H	H	H	H	-	-	L	M	M	L	M	-	-	-
CLO-5 :	Create Programs using symbolic, automata based and graphical user interface programming paradigms	3	85	80	H	H	H	H	H	-	-	L	M	M	L	M	-	-	-
CLO-6 :	Create Programs using different programming paradigms using python language	3	85	80	H	H	H	H	H	-	-	L	M	M	L	M	-	-	-

Duration (hour)		15	15	15	15	15
S-1	SLO-1	Structured Programming Paradigm	Event Driven Programming Paradigm	Parallel Programming Paradigm	Logic Programming Paradigm	Symbolic Programming Paradigm
	SLO-2	Programming Language Theory	Event Object, handler, bind	Multi-threading, Multi-Processing	First-class function, Higher-order function, Pure functions, Recursion	Symbolic Maths, algebraic manipulations, limits, differentiation, integration, series
S-2	SLO-1	Bohm-Jacopini structured program theorem	Keypress events, Mouse events	Serial Processing, Parallel Processing	Packages: Kanren, SymPy	SymPy usage for symbolic maths
	SLO-2	Sequence, selection, decision, iteration, recursion	Automatic events from a timer	Multiprocessing module in Python	PySWIP, PyDatalog	Equation Solving, Matrices
S-3	SLO-1	Other languages: C, C++, Java, C#, Ruby	Other languages: Algol, Javascript, Elm	Process class, Pool class	Other languages: Prolog, ROOP, Janus	Other languages: Aurora, LISP, Wolfram
	SLO-2	Demo: Structured Programing in Python	Demo: Event Driven Programming in Python	Demo: Parallel Programming in Python	Demo: Logic Programming in Python	Demo: Symbolic Programming in Python
S 4-5	SLO-1 SLO-2	Lab 1: Structured Programming	Lab 4: Event Driven Programming	Lab 7: Parallel Programming	Lab 10: Logic Programming	Lab 13: Symbolic Programming
S-6	SLO-1	Procedural Programming Paradigm	Declarative Programming Paradigm	Concurrent Programming Paradigm	Dependent Type Programming Paradigm	Automata Based Programming Paradigm
	SLO-2	Routines, Subroutines, functions	Sets of declarative statements	Parallel Vs Concurrent Programming	Logic Quantifier: for all, there exists	Finite State Machine, deterministic finite automation (dfa), nfa
S-7	SLO-1	Using Functions in Python	Object attribute, Binding behavior	threading, multiprocessing	Dependent functions, dependent pairs	State transitions using python-automaton
	SLO-2	logical view, control flow of procedural programming in various aspects	Creating Events without describing flow	concurrent.futures, gevent, greenlets, celery	Relation between data and its computation	Initial state, destination state, event (transition)
S-8	SLO-1	Other languages: Bliss, ChucK, Matlab	Other languages: Prolog, Z3, LINQ, SQL	Other languages: ANI, Plaid	Other Languages: Idris, Agda, Coq	Other languages: Forth, Ragel, SCXML
	SLO-2	Demo: creating routines and subroutines using functions in Python	Demo: Declarative Programming in Python	Demo: Concurrent Programming in Python	Demo: Dependent Type Programming in Python	Demo: Automata Based Programming in Python

S 9-10	SLO-1	*Lab 2: Procedural Programming*	*Lab 5: Declarative Programming*	*Lab 8: Concurrent Programming*	*Lab 11: Dependent Type Programming*	*Lab 14: Automata Programming*
	SLO-2					
S-11	SLO-1	*Object Oriented Programming Paradigm*	*Imperative Programming Paradigm*	*Functional Programming Paradigm*	*Network Programming Paradigm*	*GUI Programming Paradigm*
	SLO-2	*Class, Objects, Instances, Methods*	*Program State, Instructions to change the program state*	*Sequence of Commands*	*Socket Programming: TCP & UDP Connection oriented, connectionless*	*Graphical User Interface (GUI)*
S-12	SLO-1	*Encapsulation, Data Abstraction*	*Combining Algorithms and Data Structures*	*map(), reduce(), filter(), lambda*	*Sock_Stream, Sock_Dgram, socket(), bind(), recvfrom(), sendto(), listen()*	*Tkinter, WxPython, JPython*
	SLO-2	*Polymorphism, Inheritance*	*Imperative Vs Declarative Programming*	*partial, functools*	*Server-Client; send(), recv(), connect(), accept(), read(), write(), close()*	*WxWidgets, PyQT5*
S-13	SLO-1	*Constructor, Destructor*	*Other languages: PHP, Ruby, Perl, Swift*	*Other languages: F#, Clojure, Haskell*	*Other languages: PowerShell, Bash, TCL*	*Other languages: GTK, java-gnome*
	SLO-2	*Example Languages: BETA, Cecil, Lava Demo: OOP in Python*	*Demo: Imperative Programming in Python*	*Demo: Functional Programming in Python*	*Demo: Socket Programming in Python*	*Demo: GUI Programming in Python*
S 14-15	SLO-1	*Lab 3: Object Oriented Programming*	*Lab 6: Imperative Programming*	*Lab 9: Functional Programming*	*Lab 12: Network Programming*	*Lab 15: GUI Programming*
	SLO-2					

Learning Resources	1. *Elad Shalom, A Review of Programming Paradigms throughout the History: With a suggestion Toward a Future Approach, Kindle Edition, 2018* 2. *John Goerzen, Brandon Rhodes, Foundations of Python Network Programming: The comprehensive guide to building network applications with Python, 2nd ed., Kindle Edition, 2010* 3. *Elliot Forbes, Learning Concurrency in Python: Build highly efficient, robust and concurrent applications, Kindle Edition, 2017*	4. *Amit Saha, Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus and More, Kindle Edition, 2015* 5. *Alan D Moore, Python GUI Programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter, Kindle Edition, 2018* 6. *https://www.scipy-lectures.org/*

Learning Assessment

	Bloom's Level of Thinking	Continuous Learning Assessment (50% weightage)								Final Examination (50% weightage)	
		CLA – 1 (10%)		CLA – 2 (15%)		CLA – 3 (15%)		CLA – 4 (10%)#			
		Theory	Practice	Theory	Practice	Theory	Practice	Theory	Practice	Theory	Practice
Level 1	Remember	*20%*	*20%*	*15%*	*15%*	*15%*	*15%*	*15%*	*15%*	*15%*	*15%*
	Understand										
Level 2	Apply	*20%*	*20%*	*20%*	*20%*	*20%*	*20%*	*20%*	*20%*	*20%*	*20%*
	Analyze										
Level 3	Evaluate	*10%*	*10%*	*15%*	*15%*	*15%*	*15%*	*15%*	*15%*	*15%*	*15%*
	Create										
	Total	100 %		100 %		100 %		100 %		100 %	

CLA – 4 can be from any combination of these: Assignments, Seminars, Tech Talks, Mini-Projects, Case-Studies, Self-Study, MOOCs, Certifications, Conf. Paper etc.,

Course Designers

Experts from Industry	Experts from Higher Technical Institutions	Internal Experts
1. Mr. Sagar Sahani, Amadeus Software Labs, Bangalore, hello.sagarsahni@gmail.com	*1. Dr. Rajeev Sukumaran, IIT Madras, rajeev@wmail.iitm.ac.in*	*1. Dr. R. Annie Uthra, SRMIST*
2. Mr. Janmajay Singh, Fuji Xerox R&D, Japan, janmajaysingh14@gmail.com	*2. Prof. R. Golda Brunet, GCE, goldabrunet@gcessalem.edu.in*	*2. Dr. Christhu Raj M R, SRMIST*
		3. Ms. K. Sornalakshmi, SRMIST
		4. Mr. C. Arun, SRMIST