KA	KU	Tier	Level	Nur	PDC	Learning Outcome
	Assembly level machine					Describe how an instruction is executed in a classical von Neumann machine, with
AR	organization	2	Familiarity	2	р	extensions for threads, multiprocessor synchronization, and SIMD execution
	Assembly level machine		·			Describe instruction level parallelism and hazards, and how they are managed in
AR	organization	2	Familiarity	3	р	typical processor pipelines
	Digital logic and digital		·			Comprehend the trend of modern computer architectures towards multi-core and
AR	systems	2	Familiarity	2	р	that parallelism is inherent in all hardware systems
	Digital logic and digital		·			Explain the implications of the "power wall" in terms of further processor
AR	systems	2	Familiarity	3	р	performance improvements and the drive towards harnessing parallelism
	Functional organization					Explain basic instruction level parallelism using pipelining and the major hazards
AR		3	Familiarity	3	р	that may occur
	Interfacing and					Compare common network organizations, such as ethernet/bus, ring, switched vs.
AR	communication	2	Familiarity	4	d	routed
	Multiprocessing and					Explain the concept of interconnection networks and characterize different
AR	alternative architectures	3	Familiarity	3	d	approaches
	Multiprocessing and					Describe the differences between memory backplane, processor memory
AR	alternative architectures	3	Familiarity	5	d	interconnect, and remote memory via networks
	Multiprocessing and					Discuss the concept of parallel processing beyond the classical von Neumann model
AR	alternative architectures	3	Familiarity	1	р	
\n_	Multiprocessing and	2	Camilianity	2	_	Describe alternative architectures such as SIMD and MIMD
AR	alternative architectures Multiprocessing and	3	Familiarity	2	р	Discuss the appoint concerns that multipressessing austoms present with respect to
AR	alternative architectures	2	Comiliarity	4	_	Discuss the special concerns that multiprocessing systems present with respect to
AK	Performance	3	Familiarity	4	р	memory management and describe how these are addressed  Discuss the performance advantages that multithreading offered in an architecture
	enhancements					along with the factors that make it difficult to derive maximum benefits from this
AR	cimanocimento	2	Familiarity	5	n	approach
CN	Processing		Familiarity	-4		Describe the levels of parallelism including task, data, and event parallelism.
CIV	Processing		} <del>-</del>	` <del>`</del>	Р	Compare and contrast parallel programming paradigms recognizing the strengths
CN	1 Toccssing	2	Assessme	10	q	and weaknesses of each.
CN	Processing	}	Usage	12		Design, code, test and debug programs for a parallel computation.
OIV	Basic Rendering	<u>.</u>	osage	12	P	Describe the basic graphics pipeline and how forward and backward rendering
GV		3	Familiarity	2	р	factor in this.
"	Collaboration and	- 3	. arrimarity		P	Describe the differences between synchronous and asynchronous communication
нс	communication	3	Familiarity	1	d	beschibe the unferences between synchronous and asynchronous communication
	Defensive Programming	,				Demonstrate using a high-level programming language how to prevent a race
IAS		1	Usage	4	р	condition from occurring and how to handle an exception
IAS	Network Security		Familiarity	1		Describe the different categories of network threats and attacks
	Network Security		,			Describe virtues and limitations of security technologies at each layer of the
IAS		2	Familiarity	3	d	network stack

	Security Policy and					Understand the risks and benefits of outsourcing to the cloud
IAS	Governance	3	Familiarity	7	d	
	Web Security					Understand the browser security model including same-origin policy and threat
IAS		3	Familiarity	1	d	models in web security
	Web Security					Understand the concept of web sessions, secure communication channels such as
						TLS and importance of secure certificates, authentication including single sign-on
IAS		3	Familiarity	2	d	such as OAuth and SAML
	Web Security					Understand common types of vulnerabilities and attacks in web applications and
IAS		3	Usage	3	d	defenses against them.
IAS	Web Security	3	Usage	4	d	Understand how to use client-side security capabilities
IAS	Web Security	3	Usage	5	d	Understand how to use server-side security tools.
	Distributed Databases					Explain the techniques used for data fragmentation, replication, and allocation
IM		3	Familiarity	1	d	during the distributed database design process
	Distributed Databases					Evaluate simple strategies for executing a distributed query to select the strategy
IM		3	Assessmer	2	d	that minimizes the amount of data transfer
	Distributed Databases					Explain how the two-phase commit protocol is used to deal with committing a
IM		3	Familiarity	3	d	transaction that accesses databases stored on multiple nodes
	Distributed Databases					Describe distributed concurrency control based on the distinguished copy
IM			Familiarity	4	d	techniques and the voting method
IM	Distributed Databases	3	Familiarity	5	d	Describe the three levels of software in the client-server model
	Information Management					approaches that scale up to globally networked systems
IM	Concepts	2	Familiarity	12	d	
18.4	Information Storage and	2	Lloomo	4	لم	Perform Internet-based research
IM	Retrieval Introduction		Usage	4	d	Auticulate the examination of the Internet
NC	Introduction		Familiarity	1	d	Articulate the organization of the Internet
NC	Mobility		Familiarity	2	d	List and define the appropriate network terminology
NC NC	Networked Applications		Familiarity	2	d	Describe how wireless networks support mobile users  Implement a simple client-server socket-based application
NC	Reliable Data Delivery		Usage Familiarity	ა 1	d	Describe the operation of reliable delivery protocols
NC	Reliable Data Delivery		Usage		d d	Design and implement a simple reliable protocol
NC	Resource Allocation		•	3	d	Describe how resources can be allocated in a network
NC	Resource Allocation		Familiarity Familiarity	1	d	Describe the congestion problem in a large network
INC	Concurrency		ranimanty	2	u	Demonstrate the potential run-time problems arising from the concurrent operation
os	Concurrency	2	Usage	2	С	of many separate tasks
03	Concurrency		Usage	2	C	• •
06	Concurrency	2	Familiarity	2	_	Summarize the range of mechanisms that can be employed at the operating system
os	Concurrency	2	Familiarity	3	С	level to realize concurrent systems and describe the benefits of each
06	Concurrency	2	Familiarity	E	_	Summarize techniques for achieving synchronization in an operating system (e.g.,
OS			Familiarity	5	С	describe how to implement a semaphore using OS primitives)

	Concurrency				Describe reasons for using interrupts, dispatching, and context switching to support
os		2 Familiarity	6	С	concurrency in an operating system
	Operating System	O Familianita	4		Describe the need for concurrency within the framework of an operating system
os	Principles Overview of Operating	2 Familiarity	- 1	С	Discuss native wheel client convey distributed exercting eveters and how they differ
os	Systems	1 Familiarity	4	d	Discuss networked, client-server, distributed operating systems and how they differ
	Web Platforms		-		from single user operating systems  Design and Implement a simple web application
IPBD	Web Platforms	3 Usage	1	d	Describe the differences between Software-as-a-Service and traditional software
PBD	Web i lationiis	3 Familiarity	4	d	products
PD	Cloud Computing	3 Familiarity	1	d	4
PD	Cloud Computing	3 Familianty	!	u	Discuss the importance of elasticity and resource management in cloud computing.  Deploy an application that uses cloud infrastructure for computing and/or data
PD	Cloud Computing	3 Usage	4	d	resources
PD	Cloud Computing	3 Usage	4	u	Explain strategies to synchronize a common view of shared data across a collection
PD	Cloud Computing	3 Familiarity	2	pd	· · · · · · · · · · · · · · · · · · ·
PD	Communication and	3 Familianty	2	рu	Use mutual exclusion to avoid a given race condition
PD	Coordination	1 Usage	1	р	ose mutual exclusion to avoid a given race condition
1 0	Communication and	1 Godge		Ρ	Give an example of an ordering of accesses among concurrent activities that is not
PD	Coordination	2 Familiarity	2	С	sequentially consistent
	Communication and	2 ranmanty	_	Ŭ	Write a program that correctly terminates when all of a set of concurrent tasks
PD	Coordination	2 Usage	5	С	have completed
	Communication and	2 03agc	3	Č	Use a properly synchronized queue to buffer data passed among activities
PD	Coordination	2 Usage	6	С	ose a property synchronized quede to barrer data passed among detivities
	Communication and	3			Explain why checks for preconditions, and actions based on these checks, must
PD	Coordination	2 Familiarity	7	С	share the same unit of atomicity to be effective
	Communication and	,			Write a test program that can reveal a concurrent programming error; for example,
PD	Coordination	2 Usage	8	С	missing an update when two activities both try to increment a variable
	Communication and	3			Describe at least one design technique for avoiding liveness failures in programs
PD	Coordination	2 Familiarity	9	С	using multiple locks or semaphores
	Communication and	,			Describe the relative merits of optimistic versus conservative concurrency control
PD	Coordination	2 Familiarity	10	С	under different rates of contention among updates
	Communication and	Í			Give an example of a scenario in which blocking message sends can deadlock
PD	Coordination	2 Usage	3	d	
	Communication and				Explain when and why multicast or event-based messaging can be preferable to
PD	Coordination	2 Familiarity	4	d	alternatives
	Communication and				Use semaphores or condition variables to block threads until a necessary
PD	Coordination	3 Usage	12	С	precondition holds
PD	Distributed Systems	3 Familiarity	1	d	Distinguish network faults from other kinds of failures
	Distributed Systems				Explain why synchronization constructs such as simple locks are not useful in the
PD		3 Familiarity	2	d	presence of distributed faults

<b>DD</b>	Distributed Systems			•		Give examples of problems for which consensus algorithms such as leader election
PD	5	3	Usage	3	d	are required
	Distributed Systems					Write a program that performs any required marshalling and conversion into
						message units, such as packets, to communicate interesting data between two
PD		3	Usage	4	d	hosts
	Distributed Systems					Measure the observed throughput and response latency across hosts in a given
PD		3	Usage	5	d	network
	Distributed Systems					Explain why no distributed system can be simultaneously consistent, available, and
PD		3	Familiarity	6	d	partition tolerant
PD	Distributed Systems	3	Usage	7	d	Implement a simple server for example, a spell checking service
	Distributed Systems		<u> </u>			Explain the tradeoffs among overhead, scalability, and fault tolerance when
PD	,	3	Familiarity	8	d	choosing a stateful v. stateless design for a given service
ļ: <u>-</u>	Distributed Systems					Describe the scalability challenges associated with a service growing to
	cu .cu cyclec					accommodate many clients, as well as those associated with a service only
PD		3	Familiarity	9	d	1
D	Formal Models and	3	i aiiiiiaiity	9	u	Model a concurrent process using a formal model, such as pi-calculus
PD	Semantics	3	Usage	1	С	Model a concurrent process using a formal model, such as pi-calculus
ļ <u></u>	Formal Models and	- 3	Osage			Explain the characteristics of a particular formal parallel model
PD	Semantics	3	Familiarity	2	С	Explain the characteristics of a particular formal parallel model
ļ <u> </u>	Formal Models and	<u> </u>	arrimanty			Formally model a shared memory system to show if it is consistent
PD	Semantics	3	Usage	3	С	Torribally model a shared memory system to show it it is consistent
·	Formal Models and	1	9-			Use a model to show progress guarantees in a parallel algorithm
PD	Semantics	3	Usage	4	С	boo a model to show progress guarantees in a paramer algorithm
	Formal Models and	-	<u>J</u>			Use formal techniques to show that a parallel algorithm is correct with respect to a
PD	Semantics	3	Usage	5	С	safety or liveness property
ļ: <u>-</u>	Formal Models and	· · · · · ·	Juagu			Decide if a specific execution is linearizable or not
PD	Semantics	3	Usage	6	С	bedide if a specific execution is integrizable of flot
	Parallel Algorithms,		, ,			Compute the work and span, and determine the critical path with respect to a
	Analysis, and					parallel execution diagram
PD	Programming	2	Usage	2	р	parallel exceution diagram
	Parallel Algorithms,		Ŭ			Define "speed-up" and explain the notion of an algorithm's scalability in this regard
	Analysis, and					σ το εφουστορικό το
PD	Programming	2	Familiarity	3	р	
	Parallel Algorithms,		•			Identify independent tasks in a program that may be parallelized
	Analysis, and					, , ,
PD	Programming	2	Usage	4	р	
	Parallel Algorithms,					Characterize features of a workload that allow or prevent it from being naturally
	Analysis, and					parallelized
PD	Programming	2	Familiarity	5	р	

	Parallel Algorithms, Analysis, and					Implement a parallel divide-and-conquer and/or graph algorithm and empirically
PD	Programming	2	Usage	6	р	measure its performance relative to its sequential analog
	Parallel Algorithms,		J - 1 - 3 -			Provide an example of a problem that fits the producer-consumer paradigm
	Analysis, and					
PD	Programming	3	Familiarity	8	d	
	Parallel Algorithms,					Identify issues that arise in producer-consumer algorithms and mechanisms that
	Analysis, and					may be used for addressing them
PD	Programming	3	Familiarity	10	d	
	Parallel Algorithms,					Give examples of problems where pipelining would be an effective means of
DD	Analysis, and	2	Comiliarity	0	nd	parallelization
PD	Programming Parallel Architecture		Familiarity	9	pd	Evaluin the differences between shared and distributed mamony
PD			Familiarity	1	d	Explain the differences between shared and distributed memory
PD	Parallel Architecture		Familiarity	2	р	Describe the SMP architecture and note its key features
PD	Parallel Architecture		Familiarity	3	р	Characterize the kinds of tasks that are a natural match for SIMD machines
PD	Parallel Architecture		Familiarity	6	d	Describe the key features of different distributed system topologies
PD	Parallel Architecture		Familiarity	5	р	Describe the challenges in maintaining cache coherence
PD	Parallel Architecture		Familiarity	4	pd	Explain the features of each classification in Flynn's taxonomy
PD	Parallel Decomposition		Usage	1	р	Explain why synchronization is necessary in a specific parallel program
PD	Parallel Decomposition		Usage	2	р	Write a correct and scalable parallel algorithm
PD	Parallel Decomposition		Usage	3	р	Parallelize an algorithm by applying task-based decomposition
PD	Parallel Decomposition		Usage	4	р	Parallelize an algorithm by applying data-parallel decomposition
PD	Parallel Performance	3	Usage	1	р	Calculate the implications of Amdahl's law for a particular parallel algorithm
PD	Parallel Performance	3	Usage	4	р	Detect and correct an instance of false sharing
PD	Parallel Performance	3	Familiarity	5	р	Explain the impact of scheduling on parallel performance
PD	Parallel Performance	3	Familiarity	7	р	Explain the impact and trade-off related to power usage on parallel performance
	Parallel Performance					Describe how data distribution/layout can affect an algorithm's communication
PD		3	Familiarity	2	pd	costs
PD	Parallel Performance	3	Usage	3	pd	Detect and correct a load imbalance
PD	Parallel Performance	3	Familiarity	6	pd	Explain performance impacts of data locality
	Parallelism Fundamentals					Distinguish data races from higher level races
PD		1	Familiarity	3	р	
	Concurrency and					Write correct concurrent programs using multiple programming models.
PL	Parallelism	3	Usage	1	С	
	Concurrency and					Explain why programming languages do not guarantee sequential consistency in
PL	Parallelism	3	Familiarity	2	р	the presence of data races and what programmers must do as a result.
	Software Verification					Describe the issues and approaches to testing distributed and parallel systems.
SE	Validation	2	Familiarity	7	pd	

vs. multiple server models, motivated by real world examples (e.g., cooking recipes, lines for multiple teller machines, couple shopping for food, wash-dry-fold, etc.).  Write a simple sequential problem and a simple parallel version of the same program.  Evaluation 1 Assessmer 7 p p different problem sizes, and be able to describe the speed-ups achieved.  Fevaluation 2 performance of simple sequential and parallel versions of a program with different problem sizes, and be able to describe the speed-ups achieved.  Fevaluation 2 performance-oriented experiment, e.g., benchmark a parallel program with different data sets in order to iteratively improve its performance. Parallelism 2 p the performance-oriented experiment, e.g., benchmark a parallel program with different data sets in order to iteratively improve its performance. Parallelism 2 p the performance implications thereof.  Parallelism 2 p performance implications thereof.  Parallelism 3 p p performance implications thereof.  Parallelism 4 p p performance implications thereof.  Parallelism 5 p Parallelism 4 p p performance implications thereof.  Parallelism 6 p p performance implications thereof.  Parallelism 7 p p p performance implications thereof.  Parallelism 8 p p performance implications thereof.  Parallelism 9 p p p performance oriented experiment, e.g., benchmark a parallel program doubted experiment, e.g., benchmark a parallel program for data sets in order to iteratively improve its performance.  Parallelism 9 p performance indications thereof.  Parallelism 9 p performance indications thereof.  Parallelism 9 p performance indications thereof.  Parallelism 9 p p performance indications in order to iteratively improve its performance.  Parallelism 9 p performance indications in order to iteratively improve its performance.  Parallelism 9 p performance indications in order to iteratively impr		Computational Paradigms				Articulate the differences between single thread vs. multiple thread, single server
SF Computational Paradigms  SF Computational Paradigms  SF Computational Paradigms  SF Computational Paradigms  SF Sevaluation						
Computational Paradigms  SF 0 1 Usage 7 p program.  Computational Paradigms  SF 1 Assessmer 7 p different problem sizes, and be able to describe the speed-ups achieved.  SF Evaluation 1 Familiarity 2 p Describe Amdahl's law and discuss its limitations.  SF 1 Usage 3 p program with different data sets in order to iteratively improve its performance. Parallelism 1 Familiarity 1 p the performance implications thereof.  Parallelism 1 Familiarity 2 p performed in less elapsed time if this can be exploited.  SF Parallelism 1 Familiarity 3 p Explain other uses of parallelism, such as for reliability/redundancy of execution. Define the differences between the concepts of Instruction Parallelism.  SF Parallelism 1 Familiarity 4 p Parallelism 2 p Parallelism 3 p Explain other uses of parallelism/Multitasking, Task/Request Parallelism. Data performs simultaneously (i.e., at the same time). Explain how work can be performed in less elapsed time if this can be exploited.  SF Parallelism 4 p Parallelism 3 p Explain other uses of parallelism/Multitasking, Task/Request Parallelism. Data Parallelism 4 p Parallelism 4 p Parallelism, Thread Parallelism/Multitasking, Task/Request Parallelism. Write more than one parallel programm (e.g., one simple parallel program that manages shared resources through synchronization primitives; a simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs in terms (simple parallel program that performs size and number of resources.  SF Software Design 2 Familiarity 10 x Common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database management, or for secure communication and access.						recipes, lines for multiple teller machines, couple shopping for food, wash-dry-fold,
SF Computational Paradigms  I Assessmer 7 p program. Evaluation SF Evaluation 1 Familiarity 2 p Describe Amdahl's law and discuss its limitations. Design and conduct a performance-oriented experiment, e.g., benchmark a parallel SF Evaluation 1 Familiarity 2 p Describe Amdahl's law and discuss its limitations. Design and conduct a performance-oriented experiment, e.g., benchmark a parallel SF Parallelism 1 Familiarity 1 p Describe Amdahl's law and discuss its limitations. Design and conduct a performance-oriented experiment, e.g., benchmark a parallel SF Parallelism 1 Familiarity 1 p Describe Amdahl's law and discuss its limitations. Design and conduct a performance-oriented experiment, e.g., benchmark a parallel SF Parallelism 1 Familiarity 2 p Describe Amdahl's law and discuss its limitations. Design and conduct a performance-oriented experiment, e.g., benchmark a parallel SF Describe Amdahl's law and discuss its limitations. Describe Amdahl's law and discuss	SF		1 Familiarity	3	р	etc.).
Computational Paradigms  SF		Computational Paradigms				Write a simple sequential problem and a simple parallel version of the same
SF Evaluation   1 Assessmen   7	SF		1 Usage	7	р	program.
SF Evaluation Design and conduct a performance-oriented experiment, e.g., benchmark a parallel program with different data sets in order to iteratively improve its performance. For a given program, distinguish between its sequential and parallel execution, and be performance implications thereof.  Parallelism 1 Familiarity 2 performed in less elapsed time if this can be exploited.  SF Parallelism 1 Familiarity 3 performed in less elapsed time if this can be exploited.  SF Parallelism 1 Familiarity 4 performed in less elapsed time if this can be exploited.  SF Parallelism 2 performed in less elapsed time if this can be exploited.  SF Parallelism 4 performs a parallelism, Data so for reliability/redundancy of execution. Define the differences between the concepts of Instruction Parallelism, Data Parallelism, Data Parallelism, Data Parallelism events and operations can take place simultaneously (i.e., at the same time). Explain how work can be performed in less elapsed time if this can be exploited.  SF Parallelism 4 performed in less elapsed time if this can be exploited.  SF Parallelism 5 performed in less elapsed time if this can be exploited.  SF Parallelism 6 performed in less elapsed time if this can be exploited.  SF Parallelism 7 performed in less elapsed time if this can be exploited.  SF Parallelism 8 performed in less elapsed time if this can be exploited.  SF Parallelism 9 performed in less elapsed time if this can be exploited.  SF Parallelism 9 performed in less elapsed time if this can be exploited.  SF Parallelism 9 performed in less elapsed time if this can be exploited.  SF Parallelism 9 performed in less elapsed time if this can be exploited.  SF Parallelism 9 performance in performs in the performs simultaneously (i.e., at the same time). Explain observable performs described in less equential performs in the performed in less elapsed time if this can be exploited.  SF Parallelism 9 per		Computational Paradigms				Evaluate performance of simple sequential and parallel versions of a program with
Evaluation  SF 1 Usage 3 p program with different data sets in order to iteratively improve its performance. For a given program, distinguish between its sequential and parallel execution, and the performance implications thereof.  Parallelism  SF 1 Familiarity 2 p performed in less elapsed time if this can be exploited.  SF Parallelism 1 Familiarity 3 p Explain other uses of parallelism, such as for reliability/redundancy of execution. Define the differences between the concepts of Instruction Parallelism, Data Parallelism  Parallelism 4 p Parallelism Write more than one parallel program (e.g., one simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallelism os performance tools to measure speed-up achieved by parallel programs in terms  SF 1 Assessmer 6 p of both problem size and number of resources.  SF Software Design 3 Usage 20 x management, or for secure communication and access.  Design and conduct a performance-oriented experiment, e.g., benchmark a parallel program with different data sets in order to iteratively inprove its performance. For a given program, distinguish between its sequential and parallel execution, and the performance inplications thereof.  Demonstrate on an execution time line that parallelism events and operations can take place simultaneous (i.e., at the same time). Explain how work can be performed. Explain how work can be performed expending and parallel execution, and the performance inplications thereof.  Demonstrate on an execution time line that parallelism events and operations can take place simultaneously (i.e., at the same time). Explain how work can be performed in less elapsed time if this can be exploited.  For a given program (ing line that parallelism events and operations can take place simultaneously (i.e., at the same time). Explain how work can be performed in less elapsed in less elapsed time if this can be exploited.  For a given program ding performed experiment (i.e.g., at the same time). Explain ho	SF		1 Assessmer	7	р	different problem sizes, and be able to describe the speed-ups achieved.
SF Parallelism  Parallelism  Parallelism  SF 1 Familiarity 1 p the performance implications thereof.  Parallelism  SF 2 parallelism  Parallelism  SF 2 1 Familiarity 2 p Demonstrate on an execution time line that parallelism events and operations can take place simultaneously (i.e., at the same time). Explain how work can be performed in less elapsed time if this can be exploited.  SF Parallelism  Parallelism  SF 2 1 Familiarity 3 p Explain other uses of parallelism, such as for reliability/redundancy of execution. Define the differences between the concepts of Instruction Parallelism, Data of the differences between the concepts of Instruction Parallelism. Write more than one parallel program (e.g., one simple parallel program in more than one parallel programing paraldigm; a simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs step-by-step pipeline processing through message passing).  SF 3 1 Assessmer 6 p processing through message passing).  Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database management, or for secure communication and access.  Paurallelism  SF Software Design 3 Usage 20 x management, or for secure communication and access.	SF	Evaluation	1 Familiarity	2	р	Describe Amdahl's law and discuss its limitations.
Parallelism  SF  1 Familiarity 1 p  Parallelism  Parallelism  Parallelism  Parallelism  Parallelism  Parallelism  1 Familiarity 2 p  Parallelism  Parallelism  SF  1 Familiarity 2 p  Parallelism  SF  1 Familiarity 4 p  Parallelism  Parallelism  Parallelism  SF  1 Familiarity 4 p  Parallelism  Parallelism  Parallelism  SF  1 Semiliarity 4 p  Parallelism  Parallelism  Parallelism  SF  1 Usage  SF  1 Usage  SF  1 Assessmer 6 p  SF  1 Assessmer 6 p  SF  Software Design  3 Usage  3 Usage  2 X  Parallelism  At the performance implications thereof.  Demonstrate on an execution time line that parallelism events and operations can take place simultaneously (i.e., at the same time). Explain how work can be exploited.  Explain other uses of parallelism, such as for reliablity/redundancy of execution.  Define the differences between the concepts of Instruction Parallelism, Data Parallelism, Multitasking, Task/Request Parallelism.  Write more than one parallel program (e.g., one simple parallel program that performs step-bry-step pipeline shared resources through synchronization primitives; a simple parallel program that performs step-by-step pipeline processing through message passing).  Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database management, or for secure communication and access.		Evaluation				Design and conduct a performance-oriented experiment, e.g., benchmark a parallel
SF Parallelism  SF Parallelism	SF		1 Usage	3	р	program with different data sets in order to iteratively improve its performance.
Parallelism  Demonstrate on an execution time line that parallelism events and operations can take place simultaneously (i.e., at the same time). Explain how work can be serviced.  SF Parallelism  SF		Parallelism				For a given program, distinguish between its sequential and parallel execution, and
take place simultaneously (i.e., at the same time). Explain how work can be SF Parallelism 1 Familiarity 2 p performed in less elapsed time if this can be exploited.  SF Parallelism 1 Familiarity 3 p Explain other uses of parallelism, such as for reliability/redundancy of execution. Define the differences between the concepts of Instruction Parallelism, Data 1 Familiarity 4 p Parallelism, Multitasking, Task/Request Parallelism. Write more than one parallel program (e.g., one simple parallel program in more than one parallel programming paradigm; a simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallelism 5 processing through message passing). Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  SE Software Design 2 Familiarity 10 x Common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database management, or for secure communication and access.	SF		1 Familiarity	1	р	
SF Parallelism  SF 1 Familiarity Parallelism Paralleli		Parallelism				· ·
SF Parallelism Parallelism Parallelism SF Parallelism SF 1 Usage 5 p processing through message passing). Use performance tools to measure speed-up achieved by parallel programs in terms Parallelism SF 1 Assessmer 6 p of both problem size and number of resources. Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database Parallelism Parallelism Parallelism Parallelism Parallelism, Data Parallelism Write more than one parallel program (e.g., one simple parallel program (e.g.						
Parallelism  Parallelism  1 Familiarity Parallelism  2 Familiarity Parallelism  Define the differences between the concepts of Instruction Parallelism, Data Parallelism/Multitasking, Task/Request Parallelism.  Write more than one parallel program (e.g., one simple parallel program in more than one parallel programming paradigm; a simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs step-by-step pipeline processing through message passing). Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design  3 Usage  2 Familiarity  10 x  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  Parallelism  Norther Multitasking, Task/Request Parallel program in more than one parallel program that performs thempore parallel pro	SF		•	2	р	
Parallelism  SF  1 Usage  5 p processing through message passing). Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  Parallelism  Nuite more than one parallel program (e.g., one simple parallel program in more than one parallel program mine more than one parallel program mine parallel program to remove common cultures a simple parallel program in more than one parallel program mine parallel program that manages shared resources through synchronization primitives; a simple parallel program that manages shared resources through synchronization primitives; a simple parallel program (e.g., one simple parallel program (e.	SF		1 Familiarity	3	р	
Parallelism  Write more than one parallel program (e.g., one simple parallel program in more than one parallel programming paradigm; a simple parallel program that manages shared resources through synchronization primitives; a simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs step-by-step pipeline  SF  1 Usage 5 p  1 Usage 5 p  1 Usage 5 p  1 Assessmer 6 p		Parallelism				· · · · · · · · · · · · · · · · · · ·
than one parallel programming paradigm; a simple parallel program that manages shared resources through synchronization primitives; a simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs step-by-step pipeline simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs step-by-step pipeline of both program that performs that manages simple parallel program that performs that performs that manages simple parallel program that performs that performs that manages and program that performs that performs t	SF		1 Familiarity	4	р	,
shared resources through synchronization primitives; a simple parallel program that performs simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs step-by-step pipeline processing through message passing).  Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design  3 Usage  2 X management, or for secure communication and access.  Pawrite a simple parallel program that performs that performs step-by-step pipeline program to remove common value as simple parallel program that performs step-by-step pipeline program to remove common value as simple parallel program that performs step-by-step pipeline program to remove common value as simple parallel program that performs step-by-step pipeline parallel program that performs step-by-step pipeline program to remove common value as simple parallel program that performs step-by-step pipeline parallel program to remove common value as simple parallel program to remove common value as simple parallel program to remove common value as buffer overflows.		Parallelism				
performs simultaneous operation on partitioned data through task parallel (e.g., parallel search terms; a simple parallel program that performs step-by-step pipeline  SF						
parallel search terms; a simple parallel program that performs step-by-step pipeline  Parallelism  SF  1 Usage  5 p  processing through message passing). Use performance tools to measure speed-up achieved by parallel programs in terms  of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among  common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication  services, for database interaction including services for remote query and database  SE Software Design  3 Usage  2 x  management, or for secure communication and access.  Powrite a simple parallel program that performs step-by-step pipeline processing through message passing).  Use performance tools to measure speed-up achieved by parallel programs in terms  of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database						
Parallelism  SF  1 Usage 5 p processing through message passing). Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design  3 Usage 20 x management, or for secure communication and access.  Pewrite a simple program to remove common vulnerabilities, such as buffer overflows, integer						
Parallelism  Use performance tools to measure speed-up achieved by parallel programs in terms of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design  3 Usage  20 x management, or for secure communication and access.  Pawrite a simple program to remove common subparabilities, such as buffer overflows, integer	0.5		4.11	_		
SF 1 Assessmer 6 p of both problem size and number of resources.  Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design 3 Usage 20 x management, or for secure communication and access.  Powrite a simple program to remove common subpersibilities, such as buffer overflows, integer	SF	Develleliens	1 Usage	5	р	, , , , , , , , , , , , , , , , , , , ,
Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design  3 Usage  20 x management, or for secure communication and access.  Payrite a simple program to remove common vulnerabilities, such as buffer overflows, integer	0.5	Parallelism	4	_		
SE Software Design  2 Familiarity  10 x common software architectures such as 3-tier, pipe-and-filter, and client-server.  Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design  3 Usage  20 x management, or for secure communication and access.  Pawrite a simple program to remove common vulnerabilities, such as buffer overflows, integer	SF		1 Assessmer	ь	р	·
Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design 3 Usage 20 x management, or for secure communication and access.  Payrite a simple program to remove common vulnerabilities, such as buffer overflows, integer						
Apply component-oriented approaches to the design of a range of software, such as using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design 3 Usage 20 x management, or for secure communication and access.  Powrite a simple program to remove common vulnerabilities, such as buffer overflows, integer	SE	Software Design	2 Familiarity	10	х	common software architectures such as 3-tier, pipe-and-filter, and client-server.
using components for concurrency and transactions, for reliable communication services, for database interaction including services for remote query and database  SE Software Design 3 Usage 20 x management, or for secure communication and access.						Apply component-oriented approaches to the design of a range of software, such as
SE Software Design  3 Usage  20 x management, or for secure communication and access.  Powrite a simple program to remove common vulnerabilities, such as buffer overflows, integer						
SE Software Design 3 Usage 20 x management, or for secure communication and access.						
Powrite a simple program to remove common vulnerabilities, such as buffer overflows, integer	SE	Software Design	3 Heade	20	v	
Hongo Program to remove definition variety and addition overhowe, integer	J.L	Contware Design		20	^	Rewrite a simple program to remove common vulnerabilities, such as buffer overflows, integer
	SE	Software Construction	<sub>3</sub> Usage	8	Х	