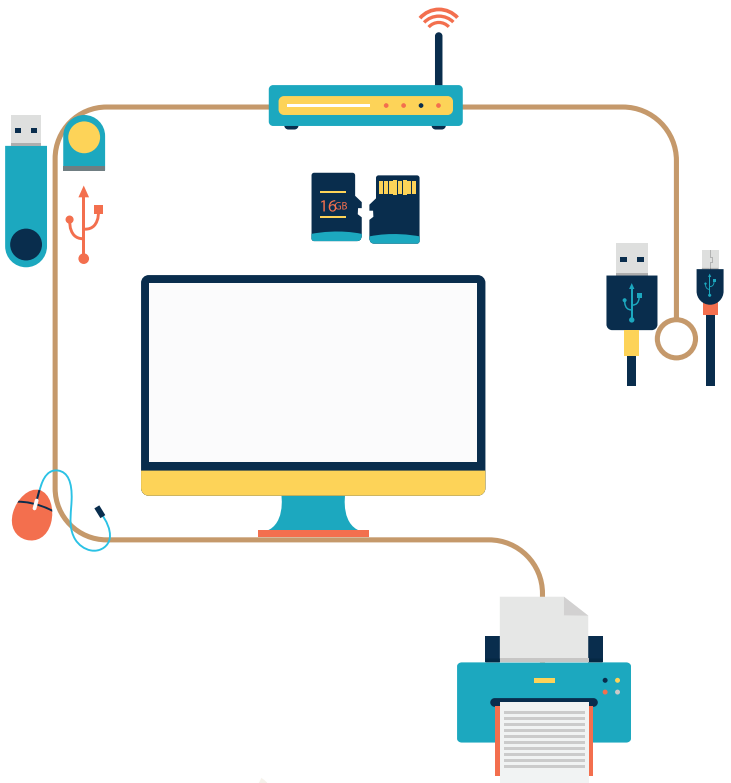
هيئة تقويم التعليم والتدريب
Education & Training Evaluation Commission

المركز الوطني للقياس
قياس
National Center for Assessment

نسخة تجريبية

LEARNING OUTEOMES

COMPUTER SCIENCE




بسم الله الرحمن الرحيم

مخرجات التعلم

تخصص علوم الحاسب

المشرف العام

د. فيصل بن عبدالله آل مشاري آل سعود

المشرف العلمي

د. عبدالله بن علي القاطعي

مدير المشروع

د. عبدالله بن صالح السعدوي

فريق العمل

١- د. عبدالعزيز ابراهيم المحيميد

٢- د. خالد عبدالله الجاسر

٣- د.عبدالله محمد المهيدب

فريق التحكيم

١- د. عبدالمحسن عيد القرني

٢- د. عبداللطيف محمد العبداللطيف

٣- د. عياد أحمد البشري

المشرف الفني

د. محمد بن علي العجيل

Introduction :

Higher Education in Saudi Arabia has witnessed a rapid development in the recent years, through inaugurating new public and private universities around the country. However, this may have an impact upon the teaching system in general and program outcomes in particular. Therefore, the Ministry of Education has endeavored to improve the quality of program outcomes in all Saudi universities. It then launched the project of learning outcomes (LOs) in Higher Education, in collaboration with the National Center for Assessment. The Bologna process which focuses primarily on LOs has been adopted widely, particularly in most European countries. Thus, this promising project will draw on the Bologna process to come up with LOs for academic programs that are being taught in Saudi universities.

LOs are basically used to ensure the quality of learning and teaching. By using them, it becomes easier to compare two different programs of the same major (i.e. benchmarking). They also help academic departments and teachers to develop course materials and determine course objectives. More importantly, they play a key role in linking teaching and learning with assessment and assisting academic programs to gain accreditation.

Furthermore, LOs have some benefits for students (stakeholders). They will provide them with the necessary information of the program they would like to join. In other words, LOs help stakeholders to know what kind of achievement they will gain by completing a certain program in cognitive (essential knowledge), behavioral (skills and abilities) and affective (attitudes, values or beliefs) domains.

Stages of the project :

This project has gone through various stages, as illustrated in Figure 1. It began with forming the main committees that will participate in this project. The National Center for Assessment ran workshops on how to write LOs and exam items based on LOs in which faculty members from various Saudi universities participated. Here are the main stages in more details.

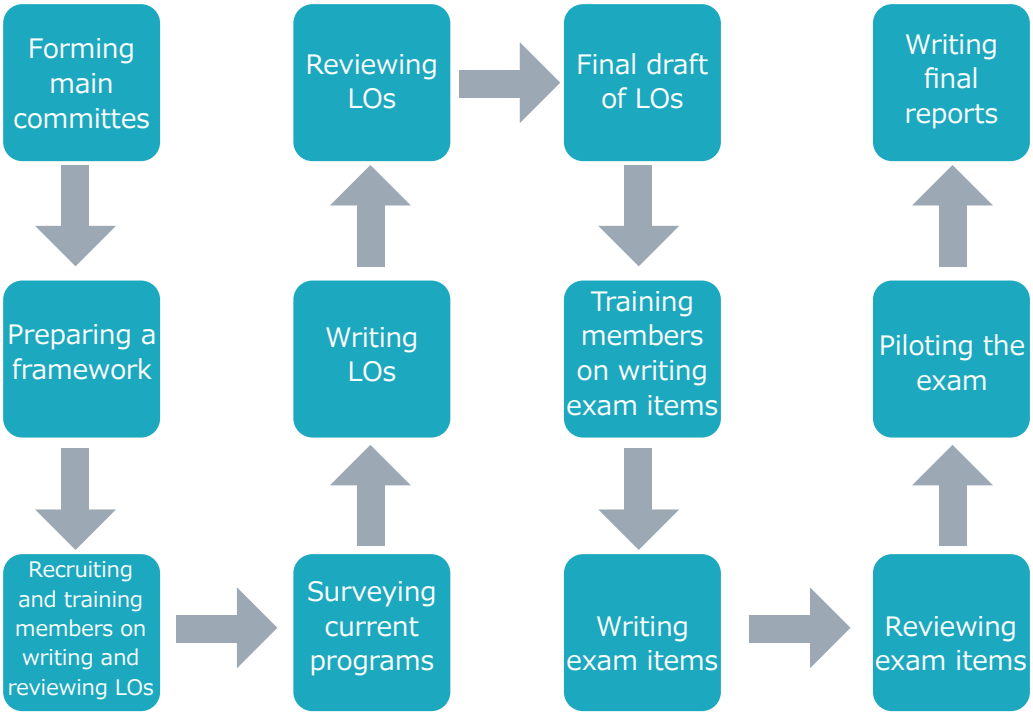


Figure 1. Stages of the LOs project

First Phase: Surveying current academic programs

This phase aims to survey the content of national and international academic programs. The objective is to establish the LOs based on these programs and identify the extent of coverage of these LOs in the academic programs in Saudi universities. The most important steps of this phase include:

1.1 Identifying LOs

A comprehensive survey has been conducted on all programs of computer science in Saudi universities, in an attempt to identify the LOs of this major.

1.2 Analyzing the content of the national programs

After collecting the content of relevant programs, a thorough analysis was done in order to identify the common components in these programs and the ones that are unique to certain programs. This procedure includes the following:

- Identifying the main components of the major in all Saudi universities.
- Determining the percentages of the main components in these programs.
- Identifying the common sub-components in these programs.
- Determining the percentages of the sub-components in these programs.

1.3 Analyzing the content of some international programs:

The previous procedure was done on the programs of the following universities:

- Arizona State University.
- Texas A&M University.
- George Mason University

A comparison was made among the components of the national and international programs in order to identify the common main and sub-components in these programs and the ones that are unique to certain programs.

1.4 Comparing the content of the national and international programs

A comparison was made among the components of the national and international programs in order to identify the common main and sub-components in these programs and the ones that are unique to certain programs.

Second Phase: Proposing the LOs of the program

This phase focuses mainly on identifying the components and their importance in the program. This procedure includes the following.

1. Defining the major accurately and comprehensively in order to determine the features that distinguish it from other similar programs.

2. Proposing the components of the program, based on the survey in the previous phase, and identifying the programs to which they are compared for benchmarking purposes.

3. Determining the importance of each component. To do so, the teaching hours of each component in the program have been calculated.

4. Dividing the main components into sub-components.

5. Identifying the importance of the sub-components, as is illustrated in Table 1.

6. Defining the main components and sub-components of the program on which the LOs will be based.

Table 1. Percentages of main components and sub-components of Computer Science

Main component	%	Sub-components	%
Algorithm	14	Basic algorithmic analysis	8
		Fundamental computing algorithms	6
Architecture and Organization	12	Digital logic and digital systems	3
		Machine level representation of data	2
		Assembly level machine organization	5
		Memory system organization and architecture	2
Discrete Structures	14	Foundations of Discrete Mathematics	6
		Basic Logic and Proof Techniques	4
		Discrete probability	4
Information Management	8	Database systems	5
		Data modelling	3

Main component	%	Sub-components	%
Net-Centric Computing	6	Communication and networking	9
	3	Network security	
Operating Systems	3	Operating system principles	8
	2	Concurrency	
	1	Scheduling and dispatch	
	2	Memory management	
Programming and Programming Languages	6	Programming constructs and problem-solving	22
	7	Data Structures	
	1	Overview of programming languages	
	4	Introduction to language translation	
	4	Object-oriented programming	
Software Engineering	2	Software Processes	13
	3	Software design	
	3	Software requirements and specifications	
	2	Software Validation and Evolution	
	3	Software project management	
Total			100

Third Phase: Writing LOs:

When writing LOs, the following points have been taken into consideration:

1. Drawing on the criteria of writing LOs reported in the literature, e.g. using measurable verbs.
2. Covering Bloom's Taxonomy levels, particularly knowledge, application and analysis.
3. Determining the target content, taking into account the division of the program (i.e. main components, sub-components and LOs) as well as the identification importance of program main and sub-components.

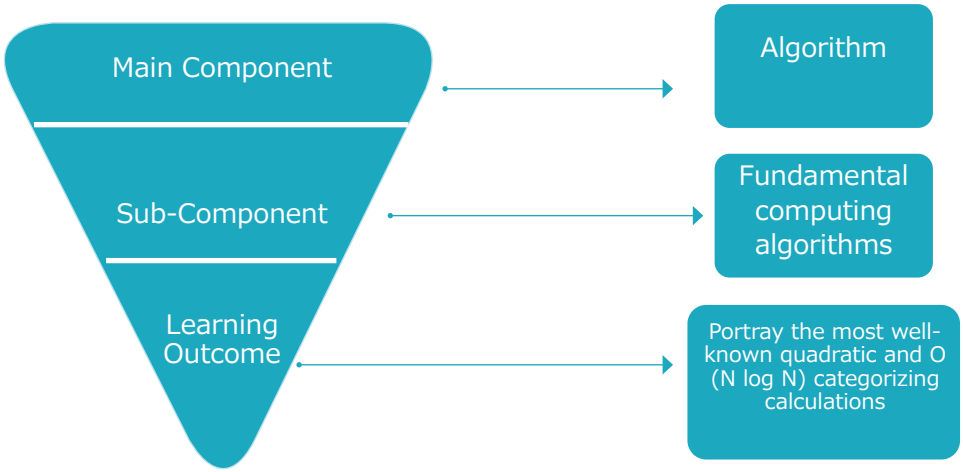


Figure 2. Illustration of the program division into main components, sub-components and LOs

Fourth Phase: Reviewing LOs:

To ensure the quality of the writing process and the use of criteria of writing LOs, the review process went through three stages:

1. Program experts

   Three experts of the program were recruited for reviewing the LOs. They were trained on how to assess LOs.

2. National universities

   A draft of the LOs was sent to all Saudi universities, in an attempt to get feedback from the faculty members of Computer Science Department in these universities. This was a very crucial step as it showed us to what extent the LOs covered the major and whether the importance of main components and sub-components was determined properly.

3. Electronic review

   The draft of the LOs was also posted on the website of the National Center for Assessment, in an attempt to get feedback from experts of Computer Science everywhere. Then, it was advertised that the LOs for Computer Science were available online for review.

Fifth Phase: Revising LOs

The comments and feedback received from the review process were approved by the reviewing committee and then sent to the committee of writing LOs to revise them accordingly. After revising the LOs, the reviewing committee approved the changes that were made.

Sixth Phase: Final draft of LOs

After the revision process, the final draft of the LOs for Computer Science was written for official use in the future, as is shown in Table 2.

1-Main Component: Algorithm

Graduates are expected to understand the basics of computer science and software engineering lay in algorithms. In the present world there are some factors on which the software system is dependent; they are: 1) the selected algorithm, and 2) how much effective and apt the different levels of implementation are.

Sub-Components	Learning Outcomes
1.1 Fundamental algorithmic analysis To cover the usage of asymptotic lower, upper and tight bounds of time and space complexity of algorithms like big(O), omega and theta notation so that it can be possible to describe the quantity of work done so far by an algorithm.	1) Clarify the need for using big(O), omega and theta notation for performance description.
	2) Use the above notation for giving asymptotic lower, upper and tight bounds on time-based and space complexity and determine simple algorithms complexity and timing.
	3) Find recurrence-relations that depict the complicated nature of time about recursively characterized algorithms.
	4) Find solutions of elementary recurrence-relations.
	5) Portray the idea of recursion and give samples of its utilization.
	6) Investigate the recursive capacities and methodology.
	7) Identify the time for the requirement of a recursive arrangement suitable for an issue.

Sub-Components	Learning Outcomes
1.2 Fundamental computing algorithms To cover the framework, implementation and evaluation of the most common algorithms like quadratic, O (N log N) sorting algorithms, hashing function for an application and collision-resolution algorithm for hash table. Graph algorithms and limited state machines should be applied and resolving problems with the help of these algorithms is to be started.	1) Implement different types of algorithms and applied them (savage power, covetous, partition & overcome, back-tracking, branch & bound, and heuristic) by taking example of common human behavior.
	2) Procedure for numerical estimate to take care of mathematical issues, for example, discovering the foundations of a polynomial.
	3) Portray the most well-known quadratic and O (N log N) categorizing calculations.
	4) Plan and execute a fitting hashing capacity for an application and its impact on determination algorithm.
	5) Discourse about the computational productivity of the central algorithms for categorizing, looking, and hashing.
	6) Find solutions of issues utilizing the crucial diagram algorithms, including profundity first and expansiveness first search, single-source and all-sets most brief ways, transitive conclusion, topological sort, and no less than one base spreading over tree calculation.

2. Main Component: Architecture and Organization:

Graduates are expected to make out that computing professionals should not consider the computer as a mere black box through which programs are carried out instantly. The learning zone of structural planning and association expands on frameworks essentials to build up a more profound comprehension of the equipment environment whereupon all figuring is based, and the interface it gives to higher programming layers.

Sub-Components	Learning Outcomes
2.1 Digital logic and digital systems To acquire binary, Boolean operations, digital-logic gates and digital components.	1) Exhibit a comprehension of the fundamental building block and its part in the advancement of computer modeling.
	2) Depict the elements of basic combinational and successive circuits numerically.
	3) Outline a basic circuit utilizing the essential building pieces.

Sub-Components	Learning Outcomes
2.2 Representation of Data at machine Level To comprehend the inner representation of numeric and nonnumeric characters, strings and record information.	1) Explain the reason of using different format for numerical representations.
	2) Clarify the way negative numbers are put away in sign–magnitude and twos–supplement representation.
	3) Change over numerical information starting with one configuration, then onto the next.
	4) Examine how altered length number representations influence exactness and accuracy.
	5) Depict the interior representation of nonnumeric information.
	6) Depict the interior representation of characters, strings, records, and arrays.

Sub-Components	Learning Outcomes
2.3 Assembly level machine organization To utilize an assembly- level machine association by a method for a standard low-level assembly language. They will also acquire complete learning of computer organization.	1) Clarify the association of the established von Neumann machine and its key practical units.
	2) Show how a guideline is carried out in a traditional von Neumann machine.
	3) Show how directions are spoken to at both the machine level and in the setting of a typical constructing agent.
	4) Clarify distinctive guideline configurations, for example, addresses for every direction and variable length versus settled length groups.
	5) Putting down low level computing construct project fragments.
	6) See how principal high-level state programming are executed at the machine-language level.
	7) Clarify how subroutine calls are taken care of, at the assembly level.
	8) Clarify the fundamental ideas of interferes with and I/O operations.

Sub-Components	Learning Outcomes
2.4 Memory system organization and architecture To identify of the principal kinds of memory technology and the impacts of memory inactivity on the running time.	1) Recognize the primary sorts of memory innovation.
	2) Clarify the impact of memory-latency on running time.
	3) Clarify the utilization of memory-hierarchy of importance to decrease the viable memory-latency idleness.
	4) Depict the memory-management principles.
	5) Explain the role to be played by cache and virtual memory.
	6) Clarify how a system works using cache and virtual memory.

3.Main Component: Discrete Structures

Graduates are expected to recognize that the discrete structures are the foundation of computer science. By foundation, it is suggested that a few computer scientists will be working basically on discrete structures, however, there are several parts of computer science and they need the ability to work with concepts belonging to discrete structures

Sub-Components	Learning Outcomes
3.1 Foundations of Discrete Mathematics To comprehend the essentials of discrete arithmetic, especially in the fundamental phrasing of functions, relations and sets, and the essential wording of graph theory. Additionally, to perform and compute permutations and combinations of sets and clarify the importance of the setting of the specific application. Furthermore, to explain and solve different types of essential recurrence-equations, and additionally make recurrence-equations. With respect to graph theory, graduates can show by illustration a few properties of exceptional cases of graph theory and exhibit distinctive traversal strategies for trees and diagrams.	1) Know the fundamental wording of functions, relations, and sets. 2) Perform the operations connected with sets, functions and relations. 3) Figure permutations and combinations of a set, and understand the importance in the connection of the specific application. 4) Relate suitable instances and explain the related operations and terms in the context. 5) Depict the meaning of the Master theorem. 6) Explain a scope of essential recurrence-equations. 7) Clarify an issue to make applicable recurrence-equations or to distinguish vital numbering questions. 8) Illustrate the fundamental phrasing of graph theory, and a portion of the properties and special instances of each theory. 9) Frame out the issues in software engineering utilizing graphs and trees. 10) Relate diagrams and trees to information structures, algorithms, and counting.

Sub-Components	Learning Outcomes
3.2 Basic Logic and Proof Techniques To comprehend fundamental rationale and proof strategies, especially in the formal techniques and apparatuses of typical propositional predicate-logic. Likewise portray and utilize these formal logic and rational thinking to take care of issues, for example, puzzles. As to proof strategies, each of the verification methods will be covered by the graduates and they will pick the suitable one for a given issue.	1) Apply formal techniques of typical propositional and predicate-logic.
	2) Depict the process through which the formal apparatuses of typical rationale are utilized to model algorithms and genuine circumstances.
	3) Use formal logical proofs and consistent thinking to take care of issues, for example, puzzles.
	4) Depict the significance and restrictions of predicate-logic.
	5) Design the fundamental structure of every verification method (direct evidence, evidence by contradiction, and induction) and provide illustrations of every one.
	6) Decide which kind of verification is best for a given issue.
	7) Relate the thoughts of mathematical induction to recursion and recursively characterized structures.
	8) Distinguish the distinction in the middle of numerical and strong induction and provide illustrations of the correct utilization of each numerical and strong induction.

Sub-Components	Learning Outcomes
3.3 Discrete probability To comprehend the probabilities of occasions and yearnings of arbitrary variables for elementary issues, for example, diversions of shot are to be calculated. Likewise distinctive sorts of hypotheses, for example, binomial and Bayes to free and ward occasions are to be utilized and applied. Graduates will figure out the technique of utilizing the devices of probability to solve issues, for example, the Monte Carlo technique, the common case analysis of algorithms and hashing.	1) Compute probabilities of occasions and yearnings of arbitrary variables for basic issues, for example, games of chance. 2) Distinguish between the dependent and independent events. 3) Apply both the binomial theorem to independent occasions, and the Bayes theorem to dependent occasions. 4) Apply the instruments of possibility to take care of issues, for example, the Monte-Carlo technique, the normal case analysis of algorithms, and hashing.

4.Main Component: Information Management

Graduates are expected to understand that Information Management covers the capture, digitization, representation, organization, transformation and presentation of information.

Sub-Components	Learning Outcomes
4.1 Database systems To recognize DBMS capacities and their functions and comprehend the idea and structural architecture of DBMS. Likewise, to apply the inquiry language and clarify the measures of proficiency and viability. Furthermore, to pick and utilize a few technical answers to get over the issues identified with data protection, trustworthiness, security, and safeguarding.	1) Clarify the attributes that recognize the database approach from the conventional methodology of programming with data-files.
	2) Refer to the essential objectives, functions, models, segments, applications, and social effect of database frameworks.
	3) Clarify every part of a database framework (which specified in LO2) and provide instances of their utilization.
	4) Recognizing key DBMS works and depict their functions in a database-system.
	5) Clarify the data-independence concept and its significance in a database-system.
	6) Utilize a question dialect to inspire data from a database.
	7) Clarify measures of proficiency by measuring throughput and response-time and measure adequacy using precision and recall methods.
	8) Depict a few technical answers for the issues identified with data protection, reliability, and security.

Sub-Components	Learning Outcomes
4.2 Data modeling To comprehend the diverse sorts of data-models and their uses.	1) Make data vs. knowledge comparisons and contrast.
	2) Sort information models in light of the different ideas that depict the database structure- that is, calculated information model, physical information model, and representational information model.
	3) Depict the modelling ideas and documentation of the substance relationship model and UML, and incorporate their utilization in information modelling.
	4) Depict the principle ideas of the OO model, for example, object character, type constructors, epitome, inheritance, polymorphism, and versioning.
	5) Characterize the essential wording utilized as a part of the rational information model.
	6) Depict the essential standards of the relational information model.
	7) Represent the demonstrating ideas and documentation of the relational information model.
	8) Scrutinize/protect a little to medium-size data application as to its agreeable genuine user data need.

5. Main Component: Net-Centric Computing

Graduates are expected to understand that the Internet and computer networks are now ubiquitous and a growing number of computing activities strongly depend on the correct operation of the underlying network. Networks, both fixed and mobile, are a key part of the computing environment of today and tomorrow.

Sub-Components	Learning Outcomes
5.1 Communication and networking To comprehend the essential information about data communication and systems administration and comprehend the general standards and ideas of local area network, wide zone network and Internet technology.	1) Observe the development of early networks and the Internet.
	2) Clarify the hierarchical, layered structure of a regular network building design.
	3) Depict developing technologies in the net-driven computing region and evaluate their present abilities, impediments, and close term potential.
	4) Examine essential network norms in their historical setting.
	5) Depict the obligations of the initial four layers of the ISO model.
	6) Observe the contrasts between circuit switching and packet switching alongside the advantages and disadvantages of each.
	7) Clarify how a system can identify and right transmission errors.
	8) Represent the way a packet is routed over the Internet.
	9) Introduce a basic system with two clients and a single server utilizing standard host-design software devices, for example, DHCP.

Sub-Components	Learning Outcomes
5.2 Network security To comprehend the diverse sorts of dangers to data resources and find out the countermeasures and their limits. Additionally, to comprehend standards of data frameworks' security, including authentication, cryptography, identification and access-control models/mechanisms, multi-level database security, and Internet security.	1) Discuss the fundamental ideas of public key encryption.
	2) Describe how public key encryption functions.
	3) Recognize the difference between private and public key encryption.
	4) Review well known authentication protocols.
	5) Create and convey a PGP key match and utilize the PGP bundle to send an encrypted email message.
	6) Condense the abilities and confinements of the method for encryption that are advantageously accessible to the overall public.

6. Main Component: Operating Systems.

Graduates are expected to aware of the scope of an operating system that not only describes the features of hardware but also helps different users to interact with each other. The points that will be covered within this framework include the connections between different networks and operating systems and understanding the differentiating features between user and kernel mode and also formulating standard operating procedures with regards to the structure and function of the operating system.

Sub-Components	Learning Outcomes
6.1 Operating system principles To gain a complete understanding of the concepts that form the very basis of operating systems like compilation of data in memory, dealing with both internal and external occurrences, the processing to which different software programs are subjected to and regulation of different events.	1) Recapitulate the events in the life history of operating systems beginning from the batch systems to recent technologically advanced systems that allow many users to work on it simultaneously.
	2) Understand the problems and security issues associated with operating systems and discuss the safety features that can be used to counter these issues.
	3) Develop an insight into the impact of internet and increased public access to software on the way operating systems.
	4) Analysis the logical-layer concept.
	5) Give points in favor of developing abstract layers in a hierarchal manner.
	6) Stress on the importance of the role played by middleware and API.
	7) Give a detailed description of the use of computing resources by application software and its regulation by systems software.
	8) Give differentiating points between user and kernel-mode within an operating system.
	9) Provide the pros and cons of interrupt processing.
	10) Describe the different applications of the driver I/O queue and a device list.

Sub-Components	Learning Outcomes
6.2 Concurrency To gain an insight into the most commonly used theories in context of concurrency modeling systems.	1) Explain why concurrency is essential for the successful functioning of an operating system.
	2) Give real life examples of the situations where many different programs are run simultaneously.
	3) Give a brief description of the different ways that can help the operating systems to function efficiently within concurrent systems and explain the advantages offered by each of them.
	4) List the different steps in which a task can be broken down and the data structure set up required to run many tasks simultaneously.
	5) Describe the different methods that can be used to find solutions to the problem of mutual exclusion developing in an operating system.
	6) Explain the causes that necessitate the use of interrupt signals that can stop a task, order the central processing unit to shift from one process to another and dispatching to ensure that many users can work on the operating systems at the same time.
	7) Explain the problem that needs to be sorted out by using diagrams that describe system behavior in form of state and transition diagrams.
	8) Explain how different data arrangements in form of stacks where insertions and deletions are made at one end only and queues where insertions are made at the rear end and deletions are made at the top help various users to access operating systems at the same time.
	9) Provide the possible reasons for a deadlock.

Sub-Components	Learning Outcomes
6.3 Scheduling and dispatch To gain an understanding of the role played by scheduling, give a detailed description of the interrelationship between application domains and scheduling algorithms, and discuss how the basic concept of logic that is the main concept behind scheduling algorithms can be applied in different fields like network and program scheduling, device I/O, and in sectors that are not directly computer based.	1) Analyze the different algorithms used for deciding which task has to be completed now and which one can be interrupted in an operating system and assigning them priority, comparing their output and apply fair share schemes, also give differentiating features between these algorithms.
	2) Discuss the interrelationship between application domains and scheduling algorithms.
	3) Give a detailed description of different types of processors which are classified on basis of the frequency of tasks performed and are divided into high level scheduler or medium term scheduler or short term scheduler and I/O.
	4) Identify the differences between threads and processes.
	5) Discuss the dynamic and static approaches to real time scheduling and identify differentiating features between the two approaches.
	6) Provide the reasons that make deadline and scheduling mandatory.
	7) Discuss how the basic concept of logic that is the main concept behind scheduling algorithms can be applied in different fields like network and program scheduling, device I/O and in sectors that are not directly computer based.

Sub-Components	Learning Outcomes
6.4 Memory management To develop an understanding about the concept of memory hierarchy and cost performance trade off. Students are also expected to understand how virtual memory in which data is temporarily transferred to disk storage in case of limited memory space and how software and hardware ensure this transfer.	1) Give a detailed description of trade-offs that help in enhancing one aspect of computing while compromising with another aspect and memory hierarchy.
	2) Discuss the principles behind virtual memory in which data is temporarily transferred to disk storage in case of limited memory space and how software and hardware ensure this transfer.
	3) Explain virtual memory in terms of storage of data in cache, dividing computer memory into smaller segments and retrieval of data from the storage space.
	4) Explain the impact of memory size (main memory, cache memory, auxiliary memory) and speed of processor on tradeoffs.
	5) Explain the various methods available to assign memory to tasks; and provide the advantages of each method.
	6) Explain why cache memory is used.
	7) Identify the different features and relative pros and cons of segmentation techniques and contrast paging.
	8) Explain the causes responsible for development of thrashing and the various methods that can help in identifying and solving this problem.
	9) Give a detailed analysis of all the methods used for memory portioning namely overlays, swapping and placement, and replacement policies.

7. Main Component: Programming and Programming Languages.

Graduates are expected to develop an insight into the use of programming languages which are utilized by programmers as a tool to explain concepts and develop algorithms and solutions

Sub-Components	Learning Outcomes
7.1 Programming constructs and problem-solving To understand the basic principles on the basis of which a computer programmer is able to find a solution to a given problem. To learn the technique of breaking down a given problem to smaller sets by using structured decomposition, utilize pseudo codes or programming language to assess algorithms on parameters of performance and applicability, and find any chinks in the algorithms.	1) Conduct an in-depth analysis of the role played by basic principles of programming in the functioning of simple programs.
	2) Work on improving short programs that are based on standard and iterative structure and functions.
	3) Implement programs that can incorporates the basic building blocks of programming like simple calculations, basic I/O, standard conditional and iterative structures and the definition of functions.
	4) Decide which iteration and conditional design will be well suited to the programming problem in hand.
	5) Break down a program into smaller blocks by using the method of structural or functional decomposition.
	6) Explain how parameter passing can be assessed.
	7) Give a detailed description of the role played by algorithms in finding solutions.
	8) List the mandatory requirements that a good algorithm must fulfill.
	9) Assess, apply and improve a problem solving algorithm with the help of pseudo code or programming language.
	10) Give a detailed description of the methods used for debugging.

Sub-Components	Learning Outcomes
7.2 Data Structures To achieve expertise in certain important concepts related to computer programming namely functions and procedural programming, algorithms designed to sort, search, and control structure, recursion, linked lists, binary trees and data arrangements (like stacks and queues).	1) Explain how older data arrangements and in built data structures function and the different applications of each of them.
	2) Explain the arrangement of data structures and their function in context of memory.
	3) List the common uses of different data structures.
	4) Use a sophisticated programming language to encode a user defined data structure.
	5) Assess data structures on the basis of performance and find improved techniques to use them in a better manner.
	6) Use the following data structures like arrays, records, strings, linked lists, queues, stacks and hash tables within the design of a program.
	7) Draw up a comparison between dynamic and static data structure arrangements in terms of performance and cost-effectiveness.
	8) Decide which data structure will be best suited for a given problem.

Sub-Components	Learning Outcomes
7.3 Overview of programming languages To study the life history of programing languages and how they have reached the level of complicated forms that are in use these days and also study the evolution of the compilation process. To compare the trade-offs between different prototypes on the basis of parameters like safety, space and time efficiency and power of expression.	1) 1. Recapitulate the events in the life history of programing languages and detail how they have achieved the form used at present.
	2) 2. Defined special characteristic of each programming prototype.
	3) 3. Compare the trade-offs between different prototypes on the basis of parameters like safety, space and time efficiency and power of expression.
7.4 Introduction to language translation To develop an understanding of the process of designing and using compilers with special emphasis placed on lexical and syntax analysis, compilers and translators, parsing, runtime environment and code generation.	1) Draw up a comparison between the interpreted and compiled execution models, giving a special emphasis on the advantages offered by each arrangement.
	2) Explain how a program takes shape, its journey from a source code to an executable code and the files that are generated in this process.
	3) Explain the distinguishing points between machine dependent and machine independent translations and how these differences can be recognized during translation.
	4) Shed light on the role played by declaration models in context of programming at large.
	5) Define a variable in terms of parameters like scope, address, and value and persistence size.
	6) Demonstrate the different types of binding, scoping, visibility and lifetime management.
	7) Explain why typing and type checking are important in context of abstraction and safety.
	8) Compare trade-offs in life time management (reference counting in comparison to garbage collection).
	9) List the differentiating features between call by reference and by value parameter passing.
	10) Explain why abstractions are important in the scheme of programing at large.

Sub-Components	Learning Outcomes
7.5 Object-oriented programming To differentiate between the different sub-paradigms related to different objects and explore the underlying principles. To introduce the latest developments in the field of object oriented language like declarative meta phasing.	1) Explain why an object oriented design is important and explain the underlying principles behind encapsulation, abstraction, polymorphism and inheritance.
	2) Formulate a simple program in an object oriented programing language, apply it, assess its function and make it error free.
	3) Explain how encapsulation and information hiding functions with the help of class mechanism.
	4) Use class hierarchy and inheritance to formulate a relationship between different objects.
	5) Describe overloading and overriding in an object oriented programming language and identify the differentiating points between them.
	6) Analyze the interrelationship between the dynamic structure of a class and static structure of a class.
	7) Explain how iterators gain access to the elements of a container.

8. Main Component: Software Engineering

Graduates are expected to understand that irrespective of the branch of computing it is a must to develop good professional conduct, adhere to deadlines, work within the budget and ensure quality work so that good software programs can be developed. So, all the sectors related to computing must follow the aforementioned principles of software engineering. All software engineers have to use and apply the required methods so that the best possible results are achieved. Software engineers usually study these principles in depth to ensure good performance

Sub-Components	Learning Outcomes
8.1 Software Processes To develop an understanding of the events in the life history of software development and the various techniques and methods required for development of software programs. To ensure a real life experience with the process of software development, undergo training in different processes and understand the mechanics of process-oriented software development.	1) Describe the systems development life-cycle focusing on the deliverables produced as a result of this development process.
	2) Chose which software development model will be best suited for application in a variety of software domains, and explain the reasons of your choice.
	3) Describe the role played by process maturity models.
	4) Draw up a comparison between the waterfall model which has been conventionally used and the incremental model, object oriented and other models.
	5) Determine where the program has reached, what is the next step in the development process and what parameters can help in defining the entire process in case of development of software programs explain.

Sub-Components	Learning Outcomes
8.2 Software design To understand the different principles and methods available for software development. To choose which method is best suited for use in development of software programs and ensure its successful application. Also, to assess software design on a component basis and whether it can be reutilized.	1) Explain the characteristics that define a good software design.
	2) Draw up a comparison between structural-analyses and object oriented design.
	3) Assess different software designs on the basis of specific software design principles and concepts.
	4) Choose which design would be best suited in the development of software and apply it.
	5) Formulate a tailor made design to be used for a medium sized product with pre-specified design requirements using a tested method of software designing (such as structured or object oriented) and employ correct design notations.
	6) Assess software designs on the basis of prescribed specifications.
	7) Assess a software design at the component level.
	8) Assess a software design and decide whether it can be reused.

Sub-Components	Learning Outcomes
8.3 Software requirements and specifications To gain professional expertise in both technical and nontechnical fields by learning engineering skills. Graduates must also be proficient in the procedures, instruments and methods used to develop software systems. The main points covered under this process include prototyping, requirement elicitation, functional and nonfunctional requirements, and requirement tracking. Also, graduates are expected to gain a real time experience in dealing with consumers and understand their specifications both in verbal and written form. Students must also understand the changes that a computer based world has brought in the lives of individual people, community as a whole and organizations and keep these changes in mind while designing the parameters of a new computer system. They must also prove their mettle as a worthy student of science by exploring sectors outside their chosen area of study.	1) Use important principles and commonly used techniques to develop the required software that can be used in a medium size software system. 2) Give a detailed description of the difficulties encountered in maintaining legacy software. 3) Give the specifications of a medium sized software system using a conventionally acceptable method (this can be in form of a written document). 4) Assess the document that gives details about the specifications of software programs on the basis of standard guidelines. 5) Translate the technical terminology used in document detailing the specifications of the software into a commonly understood language.

Sub-Components	Learning Outcomes
8.4 Software Validation and Evolution To develop an understanding of the main issues related to software development and validation and assess the effect caused by them on the entire software development process. To discuss regression testing and how it facilitates release management. Also, to describe the changes needed if the specifications of a medium sized product are altered.	1) Provide the differentiating features between program validation and program verification.
	2) Test the medium sized software through many levels and undergo many tests (unit, integration, systems, and acceptance). Enumerate the differentiating points between these tests.
	3) Formulate a test plan for a medium sized code segment and assess and implement it.
	4) Discuss how different steps in the software development process influence the entire process.
	5) Give a detailed description about the problems encountered via maintaining legacy software and stress the benefits of reverse engineering.
	6) Discuss regression testing and how it facilitates release management.
	7) Describe the changes needed if the specifications of a medium sized product are altered.
	8) Discuss how a medium sized product can be modified in order to fit the requirements of a customer request.
	9) Identify the pros and cons of reusing software programs.
	10) Discuss how a software program can be reused in a specified situation.

Sub-Components	Learning Outcomes
8.5 Software project management To understand how a project is handled and reaches completion, how the software development process works and what are the various methods that are used in this process. To understand the planning methods employed, learn about the financial aspects of project management and learn to work in mutual cooperation with other team members.	1) Develop an ability to work in a team environment displaying leadership skills and ability to handle team members.
	2) Develop a plan that must be adhered to during team meetings.
	3) Discuss the responsibilities assigned to each team member in the team working on software development and explain the reasons for making the choices.
	4) Develop an understanding about how disagreements can develop in a team environment and provide the possible advantages and disadvantages of these disagreements.
	5) Demonstrate the ability to resolve disagreements and reach a consensus.
	6) Use an ad hoc method to estimate software development effort (such as time) and compare it to an actual effort required.
	7) Draw up a comparison between different qualities tests used to monitor software products.
	8) Describe the impact of risk on a software development lifecycle.
	9) Describe different categories of risk in software systems.