

Study Guide

For

**Data Structures and Algorithm
Analysis**

Department of Computer Science



Prepared by
Mesfin Fantaye
January 2012

Title of the Study Guide: Data Structures and Algorithm Analysis

Summary of the study guide

Representing information is fundamental to computer science. The primary purpose of most computer programs is to store and retrieve information usually as fast as possible. For this reason, the study of data structures and the algorithms that manipulate them is at the heart of computer science. And that is what this course is about - helping you to understand how to structure information to support efficient processing.

This course has three primary goals. The first is to present the commonly used data structures. The second goal is to introduce the idea of tradeoffs and reinforce the concept that there are costs and benefits associated with every data structure. This is done by describing, for each data structure, the amount of space and time required for typical operations. The third goal is to teach how to measure the effectiveness of a data structure or algorithm.

To this end you are required to make your readings and programming practices as per the objectives stated in this study guide. Please make sure that you meet all the objectives listed under the respective units of this guide.

Unit 1 Review of Arrays, structures, and pointers

Unit summary

A pointer is an object that can be used to access another object. A pointer provides *indirect* access rather than *direct* access to an object. In C++ a pointer is an object that stores an address (i.e., a location in memory) where other data are stored. An address is expected to be an integer, so a pointer object can usually be represented internally as an (unsigned) int. What makes a pointer object more than just a plain integer is that we can access the datum being pointed at. Doing so is known as *dereferencing* the pointer.

The **array** is the basic mechanism for storing a collection of identically-typed objects. A different type of aggregate type is the structure, which stores a collection of objects that need not be of the same type.

General objectives

The general objectives of the unit is to guide the student gear their studies so that they will be able to explain about why arrays, structures and pointers are important; how the vector is used to implement arrays in C++; how the string is used to implement strings in C++; how basic pointer syntax and *dynamic memory allocation* are used; and how pointers, arrays, and structures are passed as parameters to functions.

Specific objectives

Pertinent to this unit, can you able to;

- explain about arrays
- declare arrays and manipulate data in arrays
- explain about “array index out of bounds”
- explain the restrictions on array processing
- pass an array as a parameter to a function
- demonstrate the knowledge of declaring C++ strings
- Examine the use of string functions to process strings
- Input data into—and output data from—a string
- Explicate the pointer data type and pointer variables

- Declare and manipulate pointer variables
- Use the address of operator and the dereferencing operator
- Declare, implement and use dynamic variables
- Explore how to use the new and delete operators to manipulate dynamic variables
- Perform pointer arithmetic
- Discover dynamic arrays
- Become aware of the shallow and deep copies of data
- Discover the peculiarities of classes with pointer member variables
- Distinguish the relationship between the address of operator and classes

Questions with answers

Q.1 The smallest element of an array's index is called its

(A) lower bound. (B) upper bound. (C) range. (D) extraction. **Ans. A**

Q.2 The extra key inserted at the end of the array is called a,

(A) End key. (B) Stop key. (C) Sentinel. (D) Transposition. **Ans. (C)**

Q.3 The largest element of an array index is called its

(A) lower bound. (B) range. (C) upper bound. (D) All of these. **Ans. (C)**

Q.4 How does an array differ from an ordinary variable?

Ans.

Array Vs. Ordinary Variable

Array is made up of similar data structure that exists in any language. Array is set of similar data types. Array is the collection of similar elements. These similar elements could be all int or all float or all char etc. Array of char is known as string. All elements of the given array must be of same type. Array is finite ordered set of homogeneous elements. The number of elements in the array is pre-specified.

An ordinary variable of a simple data type can store a single element only.

For example: *Ordinary variable*: - int a

Array: - int a[10]

Q.5 What values are automatically assigned to those array elements which are not explicitly initialized?

Ans.

Garbage values are automatically assigned to those array elements that are not explicitly initialized. If garbage class is auto then array is assumed to contain garbage values. If storage class were declared static or external then all the array elements would have a default initial value as zero.

Q.6 A two dimensional array TABLE [6] [8] is stored in row major order with base address 351. What is the address of TABLE [3] [4]?

Ans.

TABLE [6] [8]

Base address 351

Let us assume that TABLE[6] [8] is of type integer.

The formula for row major form to calculate address is

$\text{Loc}(a[i][j]) = \text{base}(a) + w [n(i - \text{lbr}) + (j - \text{lbc})]$, where 'n' no. of column in array, 'w' no of bytes per storage location, 'lbr' lower bound for row index and 'lbc' lower bound for column index.

$$\begin{aligned}\text{Loc}(\text{TABLE}[3][4]) &= 351 + 2[8(3-0) + (4-0)] \\ &= 351 + 2[24+4] \\ &= 351 + 56 \\ &= 407\end{aligned}$$

Q.7 Define the term array. How are two-dimensional arrays represented in memory? Explain how address of an element is calculated in a two dimensional array. (8)

Ans:

An *array* is a way to reference a series of memory locations using the same name. Each memory location is represented by an array element. An *array element* is similar to one variable except it is identified by an index value instead of a name. An *index* value is a number used to identify an array element.

Declaring a two dimensional array- A two dimensional array is declared similar to the way we declare a one-dimensional array except that we specify the number of elements in both dimensions. For example,
`int grades[3][4];`

The first bracket ([3]) tells the compiler that we are declaring 3 pointers, each pointing to an array. We are not talking about a pointer variable or pointer array.

Instead, we are saying that each element of the first dimension of a two dimensional array reference a corresponding second dimension. In this example, all the arrays pointed to by the first index are of the same size. The second index can be of variable size. For example, the previous statement declares a two dimensional array where there are 3 elements in the first dimension and 4 elements in the second dimension.

Two-dimensional array is represented in memory in following two ways:

1. Row major representation: To achieve this linear representation, the first row of the array is stored in the first memory locations reserved for the array, then the second row and so on.
2. Column major representation: Here all the elements of the column are stored next to one another.

In row major representation, the address is calculated in a two dimensional array as per the following formula. The address of $a[i][j] = \text{base}(a) + (i * m + j) * \text{size}$ where $\text{base}(a)$ is the address of $a[0][0]$, m is second dimension of array a and size represent size of the data type.

Q.8 What is a linear array? Explain how two dimensional arrays are represented in memory.

Ans:

An *array* is a way to reference a series of memory locations using the same name. Each memory location is represented by an array element. An *array element* is similar to one variable except it is identified by an index value instead of a name. An *index* value is a number used to identify an array element. The square bracket tells the computer that the value inside the square bracket is an index.

grades[0]

grades[1]

Representation of two-dimensional arrays in memory:-

Let grades be a 2-D array as grades[3][4]. The array will be represented in memory by a block of $3 * 4 = 12$ sequential memory locations. It can be stored in two ways:- row major wise or column major wise. The first set of four array elements is placed in memory, followed by the second set of four array elements, and so on.

Questions without answers

1. The memory address of the first element of an array is called
 - a. floor address
 - c. first address
 - b. foundation address
 - d. base address
2. The memory address of fifth element of an array can be calculated by the formula
 - a. $LOC(Array[5]) = Base(Array) + w(5 - \text{lower bound})$, where w is the number of words per memory cell for the array
 - b. $LOC(Array[5]) = Base(Array[5]) + (5 - \text{lower bound})$, where w is the number of words per memory cell for the array
 - c. $LOC(Array[5]) = Base(Array[4]) + (5 - \text{Upper bound})$, where w is the number of words per memory cell for the array
 - d. None of above
3. Which of the following data structures are indexed structures?
 - a. linear arraysb.
 - B. linked lists
 - c. both of above
 - d. none of above
4. Two dimensional arrays are also called
 - a. tables arrays
 - b. matrix arrays
 - c. both of above
 - d. none of above
5. A variable P is called pointer if
 - a. P contains the address of an element in DATA.
 - b. P points to the address of first element in DATA
 - c. P can store only memory addresses
 - d. P contain the DATA and the address of DATA
6. Which of the following data structure can't store the non-homogeneous data elements?
 - a. Arrays
 - b. Records
 - c. Pointers
 - d. None
7. Which of the following data structure store the homogeneous data elements?
 - a. Arrays
 - b. Records
 - c. Pointers
 - d. None
8. Each data item in a record may be a group item composed of sub-items; those items which are indecomposable are called
 - a. elementary items
 - c. Scalars
 - b. Atoms
 - d. all of above
9. The difference between linear array and a record is
 - a. An array is suitable for homogeneous data but hte data items in a record may have different data type
 - b. In a record, there may not be a natural ordering in opposed to linear array.

- c. A record form a hierarchical structure but a linear array does not
- d. All of above

10. Which of the following statement is false?

- a. Arrays are dense lists and static data structure
- b. data elements in linked list need not be stored in adjacent space in memory
- c. pointers store the next data element of a list
- d. linked lists are collection of the nodes that contain information part and next pointer

11. When new data are to be inserted into a data structure, but there is no available space; this situation is usually called

- a. underflow
- c. Housefull
- b. Overflow
- d. saturated

12. What is the output of the following program segment?

```
int temp[5];
for (int i = 0; i < 5; i++)
    temp[i] = 2 * i - 3;
for (int i = 0; i < 5; i++)
    cout << temp[i] << " ";
cout << endl;
temp[0] = temp[4];
temp[4] = temp[1];
temp[2] = temp[3] + temp[0];
for (int i = 0; i < 5; i++)
    cout << temp[i] << " ";
cout << endl;
```

13. Suppose list is an array of five components of type int. What is stored in list after the following C++ code executes?

```
for (int i = 0; i < 5; i++)
{
    list[i] = 2 * i + 5;
    if (i % 2 == 0)
        list[i] = list[i] - 3;
}
```

14. What is wrong with the following code?

```
int *p;
int *q;
p = new int[5];
*p = 2;
for (int i = 1; i < 5; i++)
```



```

        p[i] = p[i - 1] + i;
q = p;
delete [] p;
for (int j = 0; j < 5; j++)
    cout << q[j] << " ";
cout << endl;

```

15. What is the output of the following code?

```

int *p;
int *q;
int i;
p = new int[5];
p[0] = 5;
for (i = 1; i < 5; i++)
    p[i] = p[i - 1] + 2 * i;
cout << "Array p: ";
for (i = 0; i < 5; i++)
    cout << p[i] << " ";
cout << endl;
q = new int[5];
for (i = 0; i < 5; i++)
    q[i] = p[4 - i];
cout << "Array q: ";
for (i = 0; i < 5; i++)
    cout << q[i] << " ";
cout << endl;

```

16. a. Write a statement that declares sales to be a pointer to a pointer of type double.

b. Write a C++ code that dynamically creates a two-dimensional array of five rows and seven columns and sales contains the base address of that array.

c. Write a C++ code that inputs data from the standard input device into the array sales.

d. Write a C++ code that deallocates the memory space occupied by the two-dimensional array to which sales points.

Unit 2 Overview of ADTs, Data Structures and Algorithms

Unit summary

Solving a problem involves processing data, and an important part of the solution is the careful organization of the data. This requires that we identify the collection of data items and basic operations that must be performed on them. Such a collection of data items together with the operations on the data is called an abstract data type (commonly abbreviated as ADT). The word abstract refers to the fact that the data and the basic operations defined on it are being studied independently of how they are implemented. We are thinking of what can be done with the data, not how it is done. A data structure is the implementation for an ADT.

In the most general sense, a data structure is any data representation and its associated operations. Even an integer or floating point number stored on the computer can be viewed as a simple data structure. More commonly, people use the term “data structure” to mean an organization or structuring for a collection of data items. A sorted list of integers stored in an array is an example of such a structuring. We study data structures so that we can learn to write more efficient programs.

An **algorithm** is a clearly specified set of instructions a computer follows to solve a problem. Once an algorithm is given for a problem and determined to be correct, the next step is to determine the amount of resources, such as time and space that the algorithm will require. This step is called *algorithm analysis*.

The amount of time that any algorithm takes to run almost always depends on the amount of input that it must process. We expect, for instance, that sorting 10,000 elements requires more time than sorting 10 elements. The running time of an algorithm is thus a function of the input size. The exact value of the function depends on many factors, such as the speed of the host machine, the quality of the compiler, and in some cases, the quality of the program.

General objective

The general objective of this chapter is to set the stage for the rest of what is to follow, by presenting some higher order issues related to the selection and use of data structures. The process by which a designer selects a data structure appropriate to the task at hand will be

examined and then the role of abstraction in program design also be considered. Simple data types will be used with the ADTs they model, and how they are implemented to demonstrate ADTs.

Specific objectives

Pertinent to this unit, can you able to;

- Explain the principles of algorithm analysis,
- appreciate the significant effects of the physical medium employed
- discuss the tradeoffs between time and space requirements, or vice versa.
- Explicate the commonly used data structures, their related algorithms,
- design an algorithm that makes efficient use of the computer's resources
- discuss the approaches to solving a problem and the method of choosing between them
- describe methods for evaluating the efficiency of an algorithm or computer program
- explain the role of abstraction in program design.
- Explain the relationship between problems, algorithms, and programs
- Elucidate the Need for Data Structures
- Differentiate among the concepts type simple type, aggregate type, composite type, data item, data type, abstract data type, data structure

Questions with answers

Q.1 What is an algorithm? What are the characteristics of a good algorithm?

Ans:

An **algorithm** is “a step-by-step procedure for accomplishing some task” An algorithm can be given in many ways. For example, it can be written down in English (or French, or any other "natural" language). However, we are interested in algorithms which have been precisely specified using an appropriate mathematical formalism--such as a programming language.

Every algorithm should have the following five characteristics:

- **Input:** The algorithm should take zero or more input.
- **Output:** The algorithm should produce one or more outputs.
- **Definiteness:** Each and every step of algorithm should be defined unambiguously.

- **Effectiveness:** A human should be able to calculate the values involved in the procedure of the algorithm using paper and pencil.
- **Termination:** An algorithm must terminate after a finite number of steps.

Q.2 How do you find the complexity of an algorithm? What is the relation between the time and space complexities of an algorithm? Justify your answer with an example.

Ans:

Complexity of an algorithm is the measure of analysis of algorithm. Analyzing an algorithm means predicting the resources that the algorithm requires such as memory, communication bandwidth, logic gates and time. Most often it is computational time that is measured for finding a more suitable algorithm. This is known as time complexity of the algorithm. The running time of a program is described as a function of the size of its input. On a particular input, it is traditionally measured as the number of primitive operations or steps executed.

The analysis of algorithm focuses on time complexity and space complexity. As compared to time analysis, the analysis of space requirement for an algorithm is generally easier, but wherever necessary, both the techniques are used. The space is referred to as storage required in addition to the space required storing the input data. The amount of memory needed by program to run to completion is referred to as space complexity. For an algorithm, time complexity depends upon the size of the input, thus, it is a function of input size 'n'. So the amount of time needed by an algorithm to run to its completion is referred as time complexity.

The best algorithm to solve a given problem is one that requires less memory and takes less time to complete its execution. But in practice it is not always possible to achieve both of these objectives. There may be more than one approach to solve a same problem. One such approach may require more space but takes less time to complete its execution while the other approach requires less space but more time to complete its execution. Thus we may have to compromise one to improve the other. That is, we may be able to reduce space requirement by increasing running time or reducing running time by allocating more memory space. This situation where we compromise one to better the other is known as Time-space tradeoff.

Q.3 What do you mean by complexity of an algorithm? Explain the meaning of worst case analysis and best case analysis with an example.

Ans:

The complexity of an algorithm M is the function $f(n)$ which gives the running time and/or storage space requirement of the algorithm in terms of the size n of the input data. Frequently, the storage space required by an algorithm is simply a multiple of the data size n . Therefore, the term “complexity” shall refer to the running time of the algorithm.

We find the complexity function $f(n)$ for certain cases. The two cases one usually investigates in complexity theory are :-

- i. Worst case:- the maximum value of $f(n)$ for any possible input
- ii. Best case:- the minimum possible value of $f(n)$

If we take an example of linear search in which an integer Item is to be searched in an array Data. The complexity of the search algorithm is given by the number C of comparisons between Item and $Data[k]$.

Worst case:-

The worst case occurs when Item is the last element in the array Data or is not there at all. In both the cases, we get $C(n)=n$

The average case, we assume that the Item is present in the array and is likely to be present in any position in the array. Thus, the number of comparisons can be any of the numbers $1, 2, 3, \dots, n$ and each number occurs with probability $p = 1/n$.

$$C(n) = 1 \cdot \frac{1}{n} + 2 \cdot \frac{1}{n} + \dots + n \cdot \frac{1}{n} \\ = (n+1) / 2$$

thus the average number of comparisons needed to locate the Item in the array Data is approximately equal to half the number of elements in the Data list.

Q.4 Why do we use a asymptotic notation in the study of algorithm? Describe commonly used asymptotic notations and give their significance.

Ans:

The running time of an algorithm depends upon various characteristics and slight variation in the characteristics varies the running time. The algorithm efficiency and performance in comparison to alternate algorithm is best described by the order of growth of the running time of an

algorithm. Suppose one algorithm for a problem has time complexity as c_3n^2 and another algorithm has $c_1n^3 + c_2n^2$ then it can be easily observed that the algorithm with complexity c_3n^2 will be faster than the one with complexity $c_1n^3 + c_2n^2$ for sufficiently larger values of n . Whatever be the value of c_1 , c_2 and c_3 there will be an ' n ' beyond which the algorithm with complexity c_3n^2 is faster than algorithm with complexity $c_1n^3 + c_2n^2$, we refer this n as breakeven point. It is difficult to measure the correct breakeven point analytically, so Asymptotic notation are introduced that describe the algorithm efficiency and performance in a meaningful way. These notations describe the behavior of time or space complexity for large instance characteristics. Some commonly used asymptotic notations are:

Big oh notation (O): The upper bound for the function ' f ' is provided by the big oh notation (O). Considering ' g ' to be a function from the non-negative integers to the positive real numbers, we define $O(g)$ as the set of function f such that for some real constant $c > 0$ and some non negative integers constant n_0 , $f(n) \leq cg(n)$ for all $n \geq n_0$. Mathematically, $O(g(n)) = \{f(n): \text{there exists positive constants such that } 0 \leq f(n) \leq cg(n) \text{ for all } n, n \geq n_0\}$, we say " f is oh of g ".

Big Omega notation (Ω): The lower bound for the function ' f ' is provided by the big omega notation (Ω). Considering ' g ' to be a function from the non-negative integers to the positive real numbers, we define $\Omega(g)$ as the set of function f such that for some real constant $c > 0$ and some non negative integers constant n_0 , $f(n) \geq cg(n)$ for all $n \geq n_0$. Mathematically, $\Omega(g(n)) = \{f(n): \text{there exists positive constants such that } 0 \leq cg(n) \leq f(n) \text{ for all } n, n \geq n_0\}$.

Big Theta notation (Θ): The upper and lower bound for the function ' f ' is provided by the big oh notation (Θ). Considering ' g ' to be a function from the non-negative integers to the positive real numbers, we define $\Theta(g)$ as the set of function f such that for some real constant c_1 and $c_2 > 0$ and some non negative integers constant n_0 , $c_1g(n) \leq f(n) \leq c_2g(n)$ for all $n \geq n_0$. Mathematically, $\Theta(g(n)) = \{f(n): \text{there exists positive constants } c_1 \text{ and } c_2 \text{ such that } c_1g(n) \leq f(n) \leq c_2g(n) \text{ for all } n, n \geq n_0\}$, we say " f is oh of g ".

Q5. Explain the concept of primitive data structures.

Ans.

Primitive Data Structure

Q6. Define ADT

Abstract Data Type:- A useful tool for specifying the logical properties of data type is the abstract data type or ADT. A data type is a collection of values and a set of operation on those values. The term “ADT” refer to the basic mathematical concept that defined the data types. ADT is not concerned with implementation details at all. By specifying the mathematical and logical properties of data type or structures, the ADT is a useful guideline to implementation or a useful tool to programmers who wish to use the data type correctly.

The **value definition** defines the collection of values for the ADT and consists of 2 parts: a definition clause and a condition clause.

Operation definition: - Each operation is defined as an abstract form with 3 parts: a leader, the operational pre-condition and the post condition.

(i) Complexity of an Algorithm. (ii) The space-time trade off algorithm.

(i) Complexity of an Algorithm

An algorithm is a sequence of steps to solve a problem; there may be more than one algorithm to solve a problem. The choice of a particular algorithm depends upon following consideration:-

1) Time Complexity 2) Space Complexity

Time Complexity:- The time complexity of an algorithm is the amount of time it needs to run to completion. Some of the reasons for studying time complexity are:-

- We may be interested to know in advance whether the program will provide a satisfactory real time response.
- There may be several possible solutions with different time requirement.

Space Complexity:- The space complexity of an algorithm is the amount of memory it needs to run to completion. Some of the reasons to study space complexity are: -

- There may be several possible solutions with in different space requirement.
- To estimate the size of the largest problem that a program can solve.

(ii) The Space – Time Trade Off

The best algorithm to solve a given problem is one that requires less space in memory and takes less time to complete its execution. But in practice it is not always possible to achieve both of these objectives. There may be more than one approach to solve a problem. One approach may require more space but less time to complete its execution. The 2nd approach may require less space but takes more time to complete execution. We choose 1st approach if time is a constraint and 2nd approach if space is a constraint. Thus we may have to sacrifice one at cost of the other. That is what we can say that there exists a time space trade among algorithm.

8. Define data structure.

Data Structure is a way of representing data that contains both the data item & their relationship with each other

Questions without answers

1. Define an ADT for character strings. Your ADT should consist of typical functions that can be performed on strings, with each function defined in terms of its input and output. Then define two different physical representations for strings.
2. Define an ADT for a list of integers. First, decide what functionality your ADT should provide. Then, specify your ADT in C++ in the form of an abstract class declaration, showing the functions, their parameters, and their return types.
3. Two main measures for the efficiency of an algorithm are
 - a. Processor and memory
 - c. Time and space
 - b. Complexity and capacity
 - d. Data and space
4. The time factor when determining the efficiency of algorithm is measured by
 - a. Counting microseconds
 - c. Counting the number of statements
 - b. Counting the number of key operations
 - d. Counting the kilobytes of algorithm
5. The space factor when determining the efficiency of algorithm is measured by
 - a. Counting the maximum memory needed by the algorithm
 - b. Counting the minimum memory needed by the algorithm

- c. Counting the average memory needed by the algorithm
 - d. Counting the maximum disk space needed by the algorithm
6. Which of the following case does not exist in complexity theory
- a. Best case b. Worst case c. Average case d. Null case
7. The Worst case occur in linear search algorithm when
- a. Item is somewhere in the middle of the array
 - b. Item is not in the array at all
 - c. Item is the last element in the array
 - d. Item is the last element in the array or is not there at all
8. The Average case occur in linear search algorithm
- a. When Item is somewhere in the middle of the array
 - b. When Item is not in the array at all
 - c. When Item is the last element in the array
 - d. When Item is the last element in the array or is not there at all
9. The complexity of the average case of an algorithm is
- a. Much more complicated to analyze than that of worst case
 - b. Much more simpler to analyze than that of worst case
 - c. Sometimes more complicated and some other times simpler than that of worst case
 - d. None or above

Unit 3 Algorithm Analysis

Unit summary

This unit introduces the motivation, basic notation, and fundamental techniques of algorithm analysis. We focus on a methodology known as asymptotic algorithm analysis, or simply asymptotic analysis. Asymptotic analysis attempts to estimate the resource consumption of an algorithm. It allows us to compare the relative costs of two or more algorithms for solving the same problem. Asymptotic analysis also gives algorithm designers a tool for estimating whether a proposed solution is likely to meet the resource constraints for a problem before they implement an actual program.

The unit concludes with a brief discussion of the practical difficulties encountered when empirically measuring the cost of a program, and some principles for code tuning to improve program efficiency.

General objectives

the student is is will be made to understand the concept of a growth rate, the rate at which the cost of an algorithm grows as the size of its input grows; the concept of upper and lower bounds for a growth rate, and how to estimate these bounds for a simple program, algorithm, or problem; and the difference between the cost of an algorithm (or program) and the cost of a problem.

Specific objectives

Pertinent to this unit, can you able to;

- compare two algorithms for solving some problem in terms of efficiency
- justify the needs and benefits of **Asymptotic algorithm analysis**
- demonstrate the application of **Asymptotic algorithm analysis**
- explain the reasons for ignoring constants when estimating the growth rate for the running time or other resource requirements of an algorithm.
- identify the resources required for the algorithm and the data structure
- distinguish factors that affect the running time of a program
- identify the basic operations required by an algorithm to process an input of a certain size
- distinguish size (or the number of inputs) processed
- define the **growth rate (running time)** for an algorithm

- differentiate among **linear growth rate (running time), quadratic growth rate, and exponential growth rate**
- differentiate among **best-case, average-case and worst-case** analysis of an algorithm
- make a distinction among upper bound (big-Oh) for the growth of the algorithm's running time, the lower bound (Ω) for an algorithm and Θ Notation
- determine the running-time equation for an algorithm
- Classify Function $f(n)$ into either of $O(g(n))$, $\Omega(g(n))$, or $\Theta(g(n))$
- calculate the running time for a program
- determine the space requirements (space bounds) for the data structure
- explain the space/time tradeoff principle in algorithm design.
- Differentiate among the different techniques/methods of algorithm analysis, such as asymptotic method, empirical method, and simulation method

Questions with answers

Q.1 The complexity of multiplying two matrices of order $m \times n$ and $n \times p$ is

- (A) mnp (B) mp (C) mn (D) np **Ans:A**

Q.2 Merging 4 sorted files containing 50, 10, 25 and 15 records will take _____ time

- (A) $O(100)$ (B) $O(200)$ (C) $O(175)$ (D) $O(125)$ **Ans:A**

Q.3 The minimum number of multiplications and additions required to evaluate the polynomial

$$P = 4x^3 + 3x^2 - 15x + 45$$

is

- (A) 6 & 3 (B) 4 & 2 (C) 3 & 3 (D) 8 & 3 **Ans: C**

Q.4 $O(N)$ (linear time) is better than $O(1)$ constant time.

- (A) True (B) False **Ans:B**

Q.5 An algorithm is made up of two independent time complexities $f(n)$ and $g(n)$. Then the complexities of the algorithm is in the order of

- (A) $f(n) \times g(n)$ (B) $\text{Max}(f(n), g(n))$ (C) $\text{Min}(f(n), g(n))$ (D) $f(n) + g(n)$ **Ans:B**

Q.6 Time complexities of three algorithms are given. Which should execute the slowest for large values of N ?

- (A) $O(N^{1/2})$ (B) $O(N)$ (C) $O(\log N)$ (D) *None of these* **Ans:B**

Q.7. The complexity of linear search algorithm is

- a. $O(n)$ b. $O(\log n)$ c. $O(n^2)$ d. $O(n \log n)$ **Ans. a**

Q.8. The complexity of Binary search algorithm is

- a. $O(n)$ b. $O(\log n)$ c. $O(n^2)$ d. $O(n \log n)$ Ans. b

Q.9. The complexity of Bubble sort algorithm is

- a. $O(n)$ b. $O(\log n)$ c. $O(n^2)$ d. $O(n \log n)$ Ans. c

Q.10. The complexity of merge sort algorithm is

- a. $O(n)$ b. $O(\log n)$ c. $O(n^2)$ d. $O(n \log n)$ Ans. d

Q.11. What is the smallest value of n such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is $2n$ on the same machine.

Ans.

$$F(n) = 100n^2$$

$$G(n) = 2n$$

Value of n so that $100n^2 < 2n$

The smallest value of n such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is $2n$ on the same machine is 15.

Questions without answers

1. Programs **A** and **B** are analyzed and are found to have worst-case running times no greater than $150N \log N$ and N^2 , respectively. Which program has the better guarantee on the running time for large values of N ($N > 10,000$)?

2. In terms of N , what is the running time of the following algorithm to compute X^N :

```
double power( double x, int n )
{
    double result = 1.0;

    for( int i = 0; i < n; i++ )
        result *= x;

    return result;
}
```

3. An algorithm takes 0.5 ms for input size 100. How long will it take for input size 500 (assuming that low-order terms are negligible) if the running time is

- a. linear. b. $O(N \log N)$. c. quadratic. d. cubic.

4. An algorithm takes 0.5 ms for input size 100. How large a problem can be solved in 1 min (assuming that low-order terms are negligible) if the running time is

a. linear. b. $O(N \log N)$ c. quadratic. d. cubic.

5. Order the following functions by growth rate: N , \sqrt{N} , $N^{1.5}$, N^2 , $N \log N$, $N \log \log N$, $N \log^2 N$, $N \log(N^2)$, $2/N$, 2^N , $2^{N/2}$, 37 , N^3 , and $N^2 \log N$. Indicate which functions grow at the same rate.

6. Graph the functions $8n$, $4n \log n$, $2n^2$, n^3 , and 2^n using a logarithmic scale for the x- and y-axes. That is, if the function is $f(n)$ is y , plot this as a point with x-coordinate at $\log n$ and y-coordinate at $\log y$.

Unit 4 Introduction to list

Unit summary

We define a list to be a finite, ordered sequence of data items known as elements. “Ordered” in this definition means that each element has a position in the list. A list is said to be empty when it contains no elements. The number of elements currently stored is called the length of the list. The beginning of the list is called the head, the end of the list is called the tail. There might or might not be some relationship between the value of an element and its position in the list. For example, sorted lists have their elements positioned in ascending order of value, while unsorted lists have no particular relationship between element values and positions.

In computer programs, lists are very useful abstract data types. They are members of a general category of abstract data types called containers, whose purpose is to hold other objects.

From a theoretical point of view, a list is a homogeneous collection of elements, with a linear relationship between elements. *Linear* means that, at the logical level, each element in the list except the first one has a unique predecessor, and each element except the last one has a unique successor. (At the implementation level, a relationship also exists between the elements, but the physical relationship may not be the same as the logical one.) The number of items in the list, which we call the length of the list, is a property of a list. That is, every list has a length.

Lists can be unsorted—their elements may be placed into the list in no particular order—or they can be sorted in a variety of ways. For instance, a list of numbers can be sorted by value, a list of strings can be sorted alphabetically, and a list of grades can be sorted numerically. When the elements in a sorted list are of composite types, their logical (and often physical) order is determined by one of the members of the structure, called the key member. For example, a list of students on the honor roll can be sorted alphabetically by name or numerically by student identification number. In the first case, the name is the key; in the second case, the identification number is the key. Such sorted lists are also called *key-sorted lists*.

If a list cannot contain items with duplicate keys, it is said to have *unique* keys. This chapter deals with both unsorted lists and lists of elements with unique keys, sorted from smallest to largest key value.

General objectives

In some languages, the list is a built-in structure. In C++, while lists are provided in the Standard Template Library, the techniques for building lists and other abstract data types are so important that we guide you in this unit what you should read to design and write your own. Moreover, you will also learn how to manipulate lists using their basic operations. This unit also requires you to identify different types of lists and analyze their performance of lists.

Specific objectives

Pertinent to this unit, can you able to;

- Use the list operations to implement utility routines to do the following application-level tasks:
 - Print the list of elements
 - Create a list of elements from a file
- Implement list operations for both unsorted lists and sorted lists:
 - Create and destroy a list
 - Determine whether the list is full
 - Insert an element
 - Retrieve an element
 - Delete an element
- Explain the use of Big-O notation to describe the amount of work done by an algorithm
- Compare the unsorted list operations and the sorted list operations in terms of Big-O approximations
- Define class constructors
- Overload the relational operators less than (<) and equality (==)
- Identify and apply the phases of an object-oriented methodology
- Implement a circular linked list
- Implement a linked list with a header node, a trailer node, or both
- Implement a doubly linked list
- Distinguish between shallow copying and deep copying
- Overload the assignment operator

- Implement a linked list as an array of records
- Implement dynamic binding with virtual functions

Questions with answers

Q.1 Consider a linked list of n elements. What is the time taken to insert an element after an element pointed by some pointer?

- (A) $O(1)$ (B) $O(\log_2 n)$ (C) $O(n)$ (D) $O(n \log_2 n)$

Ans:A

Q.2 In a circular linked list

- (A) components are all linked together in some sequential manner.
 (B) there is no beginning and no end.
 (C) components are arranged hierarchically.
 (D) forward and backward traversal within the list is permitted.

Ans:B

Q.3 In a linked list with n nodes, the time taken to insert an element after an element pointed by some pointer is

- (A) $O(1)$ (B) $O(\log n)$ (C) $O(n)$ (D) $O(n \log n)$

Ans:A

Q.4 A linear collection of data elements where the linear node is given by means of pointer is called

- (A) linked list (B) node list (C) primitive list (D) None of these

Ans:A

Q.5 Which of the following operations is performed more efficiently by doubly linked list than by singly linked list?

- (A) Deleting a node whose location is given
 (B) Searching of an unsorted list for a given item
 (C) Inverting a node after the node with given location
 (D) Traversing a list to process each node

Ans. (A)

Q.6 The time required to delete a node x from a doubly linked list having n nodes is

- (A) $O(n)$ (B) $O(\log n)$ (C) $O(1)$ (D) $O(n \log n)$

Ans. (C)

Q.7 Write an algorithm to insert a node in the beginning of the linked list.

Ans:

```
/* structure containing a data part and link part */
struct node
{
    int data ;
    struct node * link ;
};
/* Following adds a new node at the beginning of the linked list */
void addatbeg ( struct node **q, int num )
{
    struct node *temp ;
    /* add new node */
    temp = malloc ( sizeof ( struct node ) ) ;
    temp -> data = num ;
    temp -> link = *q ;
    *q = temp ;
}
```

Q.8 Write an algorithm to evaluate a postfix expression. Execute your algorithm using the following postfix expression as your input: a b + c d + * f ^.

Ans .

```
To evaluate a postfix expression:
clear the stack
symb = next input character
while(not end of input)
{
    If(symb is an operand)
        Push onto the stack;
    else
    {
        pop two operands from the stack;
        result = op1 symb op2;
        push result onto the stack;
    }
    symb = next input character;
}
return (pop(stack));
```

The given input as postfix expression is: - $ab+cd+*f^{\wedge}$

Symb	Stack	Evaluation
a	a	
b	a b	
+	pop a and b Push (a + b)	(a + b)
c	(a + b) c	
d	(a + b) c d	
+	pop c and d Push (c + d)	(c + d)
*	pop (a + b) and (c + d)	(a + b) * (c + d)
f	(a + b) * (c + d) f	
$^{\wedge}$	pop (a + b) * (c + d) and f	(a + b) * (c + d) $^{\wedge}$ f

The result of evaluation is $((a + b) * (c + d))^{\wedge} f$

Q.9. Define a linked-list? How are these stored in the memory? Suppose the linked list in the memory consisting of numerical values. Write a procedure for each of the following:

- To find the maximum MAX of the values in the list.
- To find the average MEAN of the values in the list.
- To find the product PROD of the values in the list.

Ans.

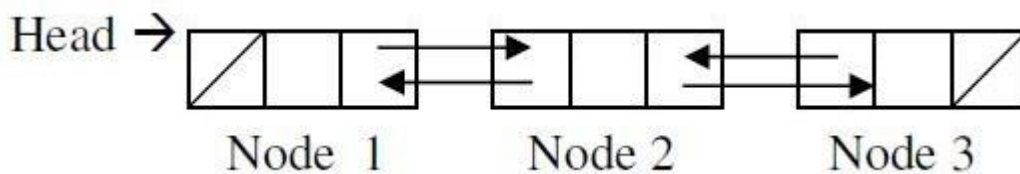
Linked List :-

A linked list is a linear collection of data elements called nodes. The linear order is given by pointer. Each node is divided into 2 or more parts. A linked list can be of the following types:-

- Linear linked list or one way list
- Doubly linked list or two way list.
- Circular linked list
- Header linked list

Representation of Linked list in Memory:-

Every node has an info part and a pointer to the next node also called as link. The number of pointers is two in case of doubly linked list. for example :



An external pointer points to the beginning of the list and last node in list has NULL. The space is allocated for nodes in the list using malloc or calloc functions. The nodes of the list are scattered in the memory with links to give linear order to the list.

```
(i)
float MAX_OF(node * start)
{
    n=start;
    max=n->k;
    while(n!= NULL)
    {
        if(max<=n->k)
            max=n->k;
    }
    return max;
(ii) float MEAN_OF(struct node * start)
{
    int h=0; struct node *n;
    n=start;
    while (n!= NULL)
    {
        s+=n->k;h++;
        n=n->next;
    }
    return s/h;
}
(iii) float PROD_OF(node *start)
{
    float p; struct node *n;
    n=start;
    while(n!= NULL)
    {
        p*=n->k;
        n=n->next;
    }
    return p;
}
```

Assume a double-linked non-circular list of integers. The order of the elements in the list is not relevant. A list element is defined as follows:

```
struct List
{
    int key;
    struct List* prev;
    struct List* next;
}
```

A list can be accessed through a pointer to the head and end, respectively. Assume two lists. List L1 is instantiated as shown in Figure 1.

```
struct List* h1;
struct List* t1;
struct List* h2;
struct List* t2;
```

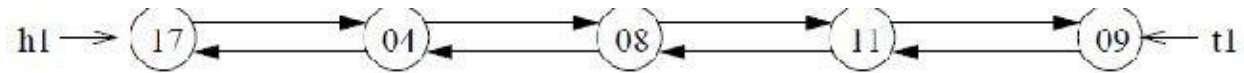


Figure 1: Original Double-linked List L1

Develop a fragment of C++ code that splits a double-linked list by removing from L1 all elements with key values less than or equal to Max. All elements less than or equal Max shall be put into a new double-linked list L2, while all other elements shall remain in the original list L1. For instance, with Max = 9 and list L1 from Figure 1, the two lists displayed in Figure 2 are produced.

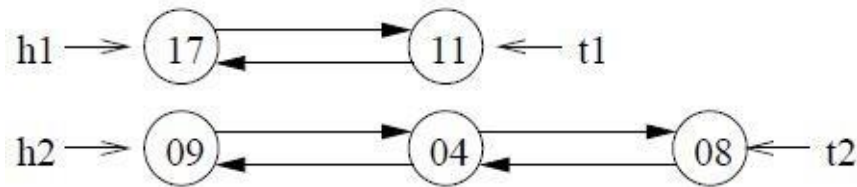


Figure 2: Resulting Double-linked Lists *L1* and *L2*

Design your algorithm so that no memory is allocated or deallocated. Illustrate your algorithm by showing the lists at different points during the computation.

Question without answers

1. The situation when in a linked list START=NULL is
 - a. underflow b. Overflow c. Housefull d. Saturated
2. Which of the following is two way list?
 - a. grounded header list b. circular header list c. linked list with header and trailer nodes d. none
3. Two linked lists contain information of the same type in ascending order. Write a module to merge them to a single linked list that is sorted.
4. Implement a circularly and doubly linked list.
5. Suppose that a singly linked list is implemented with both a header and a tail node. Describe constant-time algorithms to
 - a. insert item x before position p. b. remove the item stored at position p.

Unit 5 Stack and queue

Unit summary

A **stack** is a last in, first out (LIFO) abstract data type and linear data structure. A stack can have any abstract data type as an element, but is characterized by only three fundamental operations: *push*, *pop* and *stack top*. The push operation adds a new item to the top of the stack, or initializes the stack if it is empty. If the stack is full and does not contain enough space to accept the given item, the stack is then considered to be in an overflow state. The pop operation removes an item from the top of the stack. A pop either reveals previously concealed items, or results in an empty stack, but if the stack is empty then it goes into underflow state (It means no items are present in stack to be removed). The stack top operation gets the data from the top-most position and returns it to the user without deleting it. The same underflow state can also occur in stack top operation if stack is empty.

A stack is a *restricted data structure*, because only a small number of operations are performed on it. The nature of the pop and push operations also means that stack elements have a natural order. Elements are removed from the stack in the reverse order to the order of their addition: therefore, the lower elements are those that have been on the stack the longest.

In most high level languages, a stack can be easily implemented either through an array or a linked list. What identifies the data structure as a stack in either case is not the implementation but the interface: the user is only allowed to pop or push items onto the array or linked list, with few other helper operations.

A simple singly linked list is sufficient to implement a stack—it only requires that the head node or element can be removed, or popped, and a node can only be inserted by becoming the new head node.

A **queue** is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position and removal of entities from the front terminal position. This makes the queue a First-In-First-Out (FIFO) data structure. In a FIFO data

structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once an element is added, all elements that were added before have to be removed before the new element can be invoked. A queue is an example of a linear data structure.

Queues provide services in computer science, transport, and operations research where various entities such as data, objects, persons, or events are stored and held to be processed later. In these contexts, the queue performs the function of a buffer.

Queues are common in computer programs, where they are implemented as data structures coupled with access routines, as an abstract data structure or in object-oriented languages as classes. Common implementations are circular buffers and linked lists.

General objective

The previous unit discussed representations for lists in general. In this unit two important list-like structures called the stack and the queue are treated. Along with presenting these fundamental data structures, the other goals of the chapter are to: (1) Give examples of separating a logical representation in the form of an ADT from a physical implementation for a data structure. (2) Illustrate the use of asymptotic analysis in the context of some simple operations that you might already be familiar with. In this way you can begin to see how asymptotic analysis works, without the complications that arise when analyzing more sophisticated algorithms and data structures. (3) Introduce the concept and use of dictionaries.

Specific objectives

Pertinent to this unit, can you able to;

- Describe a stack and its operations at a logical level
- Demonstrate the effect of stack operations using a particular implementation of a stack
- Implement the Stack ADT in an array-based implementation
- Declare variables of pointer types
- Access the variables to which pointers point
- Create and access dynamically allocated data

- Explain the difference between static and dynamic allocation of the space in which the elements of an abstract data type are stored
- Use the C++ template mechanism for defining generic data types
- Define and use an array in dynamic storage
- Describe the structure of a queue and its operations at a logical level
- Demonstrate the effect of queue operations using a particular implementation of a queue
- Implement the Queue ADT using an array-based implementation
- Use inheritance to create a Counted Queue ADT
- Implement the Stack ADT as a linked data structure
- Implement the Queue ADT as a linked data structure
- Implement the Unsorted List ADT as a linked data structure
- Implement the Sorted List ADT as a linked data structure
- Compare alternative implementations of an abstract data type with respect to performance

Questions with answers

Q.1 The postfix form of the expression $(A+B)*(C*D-E)*F / G$ is

- (A) $AB + CD * E - FG / **$ (B) $AB + CD * E - F ** G /$
 (C) $AB + CD * E - * F * G /$ (D) $AB + CDE * - * F * G /$

Ans: A

Q.2 A linear list of elements in which deletion can be done from one end (front) and insertion can take place only at the other end (rear) is known as a

- (A) queue. (B) stack. (C) tree. (D) linked list.

Ans: A

Q.3 What is the postfix form of the following prefix expression $-A/B*C$DE$

- (A) $ABCDE\$*/-$ (B) $A-BCDE\$*/-$ (C) ABCED$*/-$ (D) $A-BCDE\$*/$

Ans: A

Q.4 The data structure required to evaluate a postfix expression is

- (A) queue (B) stack (C) array (D) linked-list

Ans: B

Q.5 The data structure required to check whether an expression contains balanced parenthesis is

- (A) Stack (B) Queue (C) Tree (D) Array

Ans: A

Q.6 What data structure would you mostly likely see in a nonrecursive implementation of a recursive algorithm?

(A) Stack (B) Linked list (C) Queue (D) Trees

Ans:A

Q.7 The process of accessing data stored in a serial access memory is similar to manipulating data on a

(A) heap (B) queue (C) stack (D) binary tree

Ans:C

Q.8 The postfix form of $A*B+C/D$ is

(A) $*AB/CD+$ (B) $AB*CD/+$ (C) $A*BC+/D$ (D) $ABCD+/*$

Ans:B

Q.9 Let the following circular queue can accommodate maximum six elements with the following data

front = 2 rear = 4

queue = _____; L, M, N, _____

What will happen after ADD O operation takes place?

(A) front = 2 rear = 5

queue = _____; L, M, N, O, _____

(B) front = 3 rear = 5

queue = L, M, N, O, _____

(C) front = 3 rear = 4

queue = _____; L, M, N, O, _____

(D) front = 2 rear = 4

queue = L, M, N, O, _____

Ans:A

Q.10 What is the postfix form of the following prefix $*+ab-cd$

(A) $ab+cd-*$ (B) $abc+*-$ (C) $ab+*cd-$ (D) $ab+*cd-$

Ans:A

Q.11 A stack is to be implemented using an array. The associated declarations are:

int stack[100];

int top = 0;

Give the statement to perform push operation.

Ans.

Let us assume that if stack is empty then its top has value -1.

Top ranges from 0 – 99. Let item be the data to be inserted into stack

int stack [100];

int top = 0;

int item ;

If (top == 99)

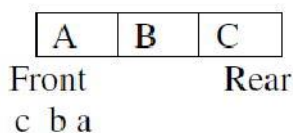
cout<< ("stack overflow");

Else

stack[top++] = item;

Q.12 Assume that a queue is available for pushing and popping elements. Given an input sequence a, b, c, (c be the first element), give the output sequence of elements if the rightmost element given above is the first to be popped from the queue.

Ans.



Q.13 A queue is a,

(A) FIFO (First In First Out) list. (B) LIFO (Last In First Out) list. (C) Ordered array. (D) Linear tree.

Ans. (A)

Q.14 Which data structure is needed to convert infix notation to postfix notation?

(A) Branch (B) Queue (C) Tree (D) Stack

Ans. (D)

Q.15 The prefix form of $A-B/(C * D ^ E)$ is,

(A) $-/*^ACBDE$ (B) $-ABCD*^DE$ (C) $-A/B*C^DE$ (D) $-A/BC*^DE$

Ans. (C)

Q.16 What is the result of the following operation Top (Push (S, X))

(A) X (B) null (C) S (D) None of these.

Ans. (A)

Q.17 The prefix form of an infix expression $p + q - r * t$ is

(A) $+pq - *rt$. (B) $- +pqr * t$.
(C) $- +pq * rt$. (D) $- + * pqrt$.

Ans. (C)

Q.18 Which data structure is used for implementing recursion?

(A) Queue. (B) Stack. (C) Arrays. (D) List.

Ans. (B)

Q.19 The equivalent prefix expression for the following infix expression $(A+B)-(C+D*E)/F*G$ is

(A) $-+AB*/+C*DEFG$ (B) $-+AB*+C*DEFG$ (C) $-/+AB*+CDE*FG$ (D) $-+AB*/+CDE*FG$

Ans. (A)

Q.20 The result of evaluating the postfix expression 5, 4, 6, +, *, 4, 9, 3, /, +, * is

(A) 600. (B) 350. (C) 650. (D) 588.

Ans. (B)

Questions without answers

1 What are circular queues? Write down routines for inserting and deleting elements from a circular queue implemented using arrays.

2 Convert the following infix expressions into its equivalent postfix expressions;

(i) $(A + B - D)/(E - F)+G$ (ii) $A*(B+D)/E - F*(G + H K)$

3. Using array to implement the queue structure, write an algorithm/program to

(i) Insert an element in the queue. (ii) Delete an element from the queue.

4. Define a stack. Describe ways to implement stack.

5. Write down any four applications of queues.

6. Which of the following name does not relate to stacks?

a. FIFO lists b. LIFO list c. Piles d. Push-down lists

18. The term "push" and "pop" is related to the

a. array b. Lists c. Stacks d. all of above

19. A data structure where elements can be added or removed at either end but not in the middle

a. Linked lists b. Stacks c. Queues d. Deque

Unit 6 Sorting

Unit summary

Putting an unsorted list of data elements into order—*sorting*—is a very common and useful operation. The goal is to come up with better, more efficient sorts. Because sorting a large number of elements can be extremely time-consuming, a good sorting algorithm is very desirable. This is one area in which programmers are sometimes encouraged to sacrifice clarity in favor of speed of execution.

How do we describe efficiency? We pick an operation central to most sorting algorithms: the operation that compares two values to see which is smaller. In our study of sorting algorithms, we relate the number of comparisons to the number of elements in the list (N) as a rough measure of the efficiency of each algorithm. The number of swaps made is another measure of sorting efficiency. In the exercises, we ask you to analyze the sorting algorithms developed in this chapter in terms of data movements.

Another efficiency consideration is the amount of memory space required. In general, memory space is not a very important factor in choosing a sorting algorithm.

The usual time versus space tradeoff applies to sorts—more space often means less time, and vice versa. Because processing time is the factor that applies most often to sorting algorithms, we consider it in detail here. Of course, as in any application, the programmer must determine goals and requirements before selecting an algorithm and starting to write code.

General objectives

The present chapter covers several standard algorithms appropriate for sorting a collection of records that fit in the computer's main memory. We review the straight selection sort and the bubble sort, then we review a more complex sorting algorithm the quick sort algorithm and introduce two additional complex sorts: merge sort and heap sort.

Specific objectives

Pertinent to this unit, can you able to;

- Design and implement the following sorting algorithms:

- Straight selection sort
- Merge sort
- Bubble sort
- Heap sort
- Insertion sort
- Radix sort
- Compare the efficiency of the sorting algorithms, in terms of Big-O complexity and space requirements
- Discuss other efficiency considerations: sorting small numbers of elements, programmer time, and sorting arrays of large data elements
- Sort on several keys
- Discuss the performances of the following search algorithms:
 - Sequential search of an unsorted list
 - Sequential search of a sorted list
 - Binary search
 - Searching a high-probability sorted list
- Define the following terms:
 - Hashing
 - Linear probing
 - Rehashing
 - Clustering
 - Collisions
- Design and implement an appropriate hashing function for an application
- Design and implement a collision-resolution algorithm for a hash table
- Discuss the efficiency considerations for the searching and hashing algorithms, in terms of Big-O notation

Questions with answers

Q1. You have to sort a list L consisting of a sorted list followed by a few “random” elements. Which of the following sorting methods would be especially suitable for such a task?

(A) Bubble sort (B) Selection sort (C) Quick sort (D) Insertion sort

Ans:D

Q.2 The number of interchanges required to sort 5, 1, 6, 2 4 in ascending order using Bubble Sort is
(A) 6 (B) 5 (C) 7 (D) 8

Ans:B

Q.3 In worst case Quick Sort has order

(A) $O(n \log n)$ (B) $O(n^2/2)$ (C) $O(\log n)$ (D) $O(n^2/4)$

Ans:B

Q.4 A sort which relatively passes through a list to exchange the first element with any element less than it and then repeats with a new first element is called

(A) insertion sort. (B) selection sort. (C) heap sort. (D) quick sort.

Ans:D

Q.5 Which of the following sorting algorithms does not have a worst case running time of $O(n^2)$?

(A) Insertion sort (B) Merge sort (C) Quick sort (D) Bubble sort

Ans:B

Q.6 The quick sort algorithm exploits _____ design technique

(A) Greedy (B) Dynamic programming (C) Divide and Conquer (D) Backtracking

Ans: C

Q.7 The total number of comparisons required to merge 4 sorted files containing 15, 3, 9 and 8 records into a single sorted file is

(A) 66 (B) 39 (C) 15 (D) 3s

Ans: D

Q.8 Which of the following sorting methods would be most suitable for sorting a list which is almost sorted

(A) Bubble Sort (B) Insertion Sort (C) Selection Sort (D) Quick Sort

Ans:A

Q.9 Quick sort is also known as

(A) merge sort (B) heap sort (C) bubble sort (D) none of these

Ans:D

Q.10 The best average behavior is shown by

(A) Quick Sort (B) Merge Sort (C) Insertion Sort (D) Heap Sort

Ans:A

Q.10 Consider that n elements are to be sorted. What is the worst case time complexity of Bubble sort?

(A) $O(1)$ (B) $O(\log 2n)$ (C) $O(n)$ (D) $O(n^2)$

Ans. (D)

Q.11 Which of the following sorting algorithm is stable

(A) insertion sort. (B) bubble sort. (C) quick sort. (D) heap sort.

Ans. (D)

Q.12 The worst case of quick sort has order

(A) $O(n^2)$ (B) $O(n)$ (C) $O(n \log 2 n)$ (D) $O(\log 2 n)$

Ans. (A)

Questions without answers

Q.1 Describe insertion sort with a proper algorithm. What is the complexity of insertion sort in the worst case?

Q.2 Which sorting algorithm is best if the list is already sorted? Why?

Q.3 What is the time complexity of Merge sort and Heap sort algorithms?

Q.4 Sort the sequence 8, 1, 4, 1, 5, 9, 2, 6, 5 by using

- a. insertion sort.
- b. Shellsort for the increments { 1, 3, 5 1.
- c. mergesort.
- d. quicksort. with the middle element as pivot and no cutoff (show all steps).
- e. quicksort, with median-of-three pivot selection and a cutoff of 3.

Q. 5. When all keys are equal, what is the running time of

- a. insertion sort.
- b. Shellsort.
- c. mergesort.
- d. quicksort.

Q.6. When the input has been sorted, what is the running time of

- a. insertion sort.
- b. Shellsort.
- c. mergesort.
- d. quicksort.

Q.7. When the input has been sorted in reverse order, what is the running time of

- a. insertion sort.
- b. Shellsort.
- c. mergesort.
- d. quicksort.

Unit 7 Searching

Unit summary

Organizing and retrieving information is at the heart of most computer applications, and searching is surely the most frequently performed of all computing tasks.

Search can be viewed abstractly as a process to determine if an element with a particular value is a member of a particular set. The more common view of searching is an attempt to find the record within a collection of records that has a particular key value, or those records in a collection whose key values meet some criterion such as falling within a range of values.

We can categorize search algorithms into three general approaches:

1. Sequential and list methods.
2. Direct access by key value (hashing).
3. Tree indexing methods.

General objective

In this unit we treat the first two approaches. Accordingly, we consider methods for searching data stored in lists. List in this context means any list implementation including a linked list or an array. Most of these methods are appropriate for sequences (i.e., duplicate key values are allowed), although special techniques applicable to sets are discussed then we will proceed to the discussion of hashing, a technique for organizing data in an array such that the location of each record within the array is a function of its key value. Hashing is appropriate when records are stored either in RAM or on disk.

Specific objectives

Pertinent to this unit, can you able to;

- Distinguish among: jump search algorithm, divide and conquer algorithm, dictionary or interpolation search algorithm, Quadratic Binary Search algorithm,
- Explain hashing algorithm
- Differentiate between hash table, hash function and slot (a position in the hash table)
- Explain collision and collision resolution policy
- Explicate open hashing (separate chaining) and closed hashing (open addressing) techniques of collision resolution
- State when it will be appropriate applying the respective collision resolution techniques

- Implement both open and closed hashing
- Determine when it is appropriate to apply hashing
- Compute the cost associated each algorithm
- Implement all the search algorithms

Questions with answers

Q.1 If h is any hashing function and is used to hash n keys in to a table of size m , where $n \leq m$, the expected number of collisions involving a particular key x is :

- (A) less than 1. (B) less than n . (C) less than m . (D) less than $n/2$.

Ans:A

Q.2 A technique for direct search is

- (A) Binary Search (B) Linear Search (C) Tree Search (D) Hashing

Ans:D

Q.3 The searching technique that takes $O(1)$ time to find a data is

- (A) Linear Search (B) Binary Search (C) Hashing (D) Tree Search

Ans:C

Q.4 The goal of hashing is to produce a search that takes

- (A) $O(1)$ time (B) $O(n^2)$ time (C) $O(\log n)$ time (D) $O(n \log n)$ time

Ans:A

Q.5 What do you mean by hashing? Explain any five popular hash functions.

Ans:

Hashing provides the direct access of record from the file no matter where the record is in the file. This is possible with the help of a hashing function H which map the key with the corresponding key address or location. It provides the key-to-address transformation.

Five popular hashing functions are as follows:

Division Method: An integer key x is divided by the table size m and the remainder is taken as the hash value. It can be defined as $H(x) = x \% m + 1$

For example, $x=42$ and $m=13$, $H(42) = 42 \% 13 + 1 = 3 + 1 = 4$

Midsquare Method: A key is multiplied by itself and the hash value is obtained by selecting an appropriate number of digits from the middle of the square. The same positions in the square must be used for all keys. For example if the key is 12345, square of this key is value 152399025. If 2 digit addresses is required then position 4th and 5th can be chosen, giving address 39.

Folding Method: A key is broken into several parts. Each part has the same length as that of the required address except the last part. The parts are added together, ignoring the last carry, we obtain the hash address for key K .

Multiplicative method: In this method a real number c such that $0 < c < 1$ is selected. For a nonnegative integral key x , the hash function is defined as $H(x) = [m(cx \% 1)] + 1$

Here, $cx \% 1$ is the fractional part of cx and $[]$ denotes the greatest integer less than or equal to its contents.

Digit Analysis: This method forms addresses by selecting and shifting digits of the original key. For a given key set, the same positions in the key and same rearrangement pattern must be used. For example, a key 7654321 is transformed to the address 1247 by selecting digits in position 1,2,4 and 7 then by reversing their order.

Q6. Explain Hash Tables, Hash function and Hashing Techniques?

Ans.

Hash Table:

A hash table is a data structure in which the location of a data item is determined directly as a function of data item itself rather than by a sequence of comparison. Under ideal condition, the time required to locate a data item in a hash table is $O(1)$ ie. It is constant and DOES not depend on the number of data items stored.

When the set of K of keys stored is much smaller then the universe U of all possible keys, a hash table require much less storage spare than a direct address table.

Hash Function

A hash function h is simply a mathematical formula that manipulates the key in some form to compute the index for this key in the hash table. Eg a hash function can divide the key by some number, usually the size of the hash table and return remainder as the index for the key. In general, we say that a hash function h maps the universe U of keys into the slots of a hash table $T[0 \dots n-1]$. This process of mapping keys to appropriate slots in a hash table is known as **hashing**.

The main reconsideration for choosing hash function is :-

- 1) It should be possible to compute it efficiently.
- 2) It should distribute the keys uniformly across the hash table i.e. it should keep the number. of collisions as minimum as possible.

Hashing Techniques:

There is variety of hashing techniques. Some of them are:-

a) Division Method:- In this method, key K to be mapped into one of the m states in the hash table is divided by m and the remainder of this division taken as index into the hash table. That is the hash function is $h(k) = k \bmod m$ where \bmod is the modules operation.

b) Multiplication Method: The multiplication method operates in 2 steps. In the 1st step the key value K is multiplied by a constant A in the range $0 < A < 1$ and the fractional part of the value obtained above is multiplied by m and the floor of the result is taken as the hash values. That is the hash function is $h(k) = [m (K A \bmod 1)]$ where “ $K A \bmod 1$ ” means the fractional part KA of $KA - [KA]$.
 $A = (5 - 1/2) = 0.6180334887$

(c) Midsquare Method:- this operates in 2 steps. In the first step the square of the key value K is taken. In the 2nd step, the hash value is obtained by deleting digits from ends of the squared values ie. K^2 . The hash function is $h(k) = s$ where s is obtained by deleting digits from both sides of K^2 .

Questions without answers

1. What are the array indices for a hash table of size 11?
2. Given the input (4371, 1323, 6173, 4199, 4344, 9679, 19891, a fixed table size of 10, and a hash function $H(X) = X \bmod 10$, show the resulting
 - a. linear probing hash table.
 - b. quadratic probing hash table.
 - c. separate chaining hash table.

Unit 8 Trees

Unit summary

The list representations of data have a fundamental limitation: Either search or insert can be made efficient, but not both at the same time. Tree structures permit both efficient access and update to large collections of data. Binary trees in particular are widely used and relatively easy to implement. But binary trees are useful for many things besides searching. Just a few examples of applications that trees can speed up include prioritizing jobs, describing mathematical expressions and the syntactic elements of computer programs, or organizing the information needed to drive data compression algorithms. Almost all operating systems store files in trees or treelike structures. Trees are also used in compiler design, text processing, and searching algorithms.

Unit general objectives

In this unit we define a general tree and discuss how it is used in a file system, examine the binary tree, implement tree operations using recursion, and nonrecursive traversal of a tree. The basic binary search tree is also discussed and a method for adding order statistics (i.e., the find^{Kth} operation) will also be seen, the three different ways to eliminate the $O(N)$ worst case (namely, the AVL tree, red-black tree, and AA-tree) are examined, the binary search implementation of the STL set and map, and use of the B-tree to search a large database quickly is also studied.

Specific objectives

Pertinent to this unit, can you able to;

- Define binary trees
- Clearly distinguish among the concepts: node, subtree, root node, edge, parent node, children node, leaf node, internal node, path, length of path, ancestor, descendant, depth of a node, height of a tree, level, full binary tree, and complete binary tree, tree traversal, enumeration
- Differentiate among preorder, postorder and inorder traversal
- Demonstrate the ability to traverse a binary tree
- Implement binary tree nodes by pointer-based binary tree node implementations and/or array-based implementation for complete binary trees.

- determining the space requirements for a given implementation (pointer-base or array-based)
- state the properties of binary search trees
- differentiate between Heaps and Priority Queues

Questions with answers

Q.1 If a node having two children is deleted from a binary tree, it is replaced by its

- (A) Inorder predecessor (B) Inorder successor (C) Preorder predecessor (D) None of the above

Ans:B

Q.2 A full binary tree with $2n+1$ nodes contain

- (A) n leaf nodes (B) n non-leaf nodes (C) $n-1$ leaf nodes (D) $n-1$ non-leaf nodes

Ans:B

Q.3 If a node in a BST has two children, then its inorder predecessor has

- (A) no left child (B) no right child (C) two children (D) no child

Ans:B

Q.4 A binary tree in which if all its levels except possibly the last, have the maximum number of nodes and all the nodes at the last level appear as far left as possible, is known as

- (A) full binary tree. (B) AVL tree. (C) threaded tree. (D) complete binary tree.

Ans:A

Q.5 The number of different directed trees with 3 nodes are

- (A) 2 (B) 3 (C) 4 (D) 5

Ans: B

Q.6 One can convert a binary tree into its mirror image by traversing it in

- (A) inorder (B) preorder (C) postorder (D) any order

Ans:C

Q.7 The complexity of searching an element from a set of n elements using Binary search algorithm is

- (A) $O(n)$ (B) $O(\log n)$ (C) $O(n^2)$ (D) $O(n \log n)$

Ans:B

Q.8 The number of leaf nodes in a complete binary tree of depth d is

- (A) $2d$ (B) $2^{d-1}+1$ (C) $2^{d+1}+1$ (D) 2^{d+1}

Ans:A

Q.9 A B-tree of minimum degree t can maximum_____ pointers in a node.

- (A) $t-1$ (B) $2t-1$ (C) $2t$ (D) t

Ans:D

Q.10 A BST is traversed in the following order recursively: Right, root, left. The output sequence will be in

- (A) Ascending order (B) Descending order (C) Bitomic sequence (D) No specific order

Ans:B

Q.11 The pre-order and post order traversal of a Binary Tree generates the same output. The tree can have maximum

- (A) Three nodes (B) Two nodes (C) One node (D) Any number of nodes

Ans:C

Q.12 A binary tree of depth “d” is an almost complete binary tree if

(A) Each leaf in the tree is either at level “d” or at level “d-1”

(B) For any node “n” in the tree with a right descendent at level “d” all the left descendents of “n” that are leaves, are also at level “d”

(C) Both (A) & (B)

(D) None of the above

Ans:C

Q.13 One of the major drawback of B-Tree is the difficulty of traversing the keys sequentially.

(A) True (B) False

Ans:A

Q.14 A characteristic of the data that binary search uses but the linear search ignores is the_____.

(A) Order of the elements of the list. (B) Length of the list.

(C) Maximum value in list. (D) Type of elements of the list.

Ans. (A)

Q.15 How many nodes in a tree have no ancestors.

(A) 0 (B) 1 (C) 2 (D) n

Ans. (B)

Q.16 In order to get the contents of a Binary search tree in ascending order, one has to traverse it in

(A) pre-order. (B) in-order. (C) post order. (D) not possible.

Ans. (B)

Q.17 In binary search, average number of comparison required for searching an element in a list if n numbers is

(A) $\log_2 n$. (B) $n / 2$. (C) n. (D) $n - 1$.

Ans. (A)

Q.18 In order to get the information stored in a Binary Search Tree in the descending order, one should traverse it in which of the following order?

(A) left, root, right (B) root, left, right (C) right, root, left (D) right, left, root

Ans. (C)

Questions without answers

1 What is a Binary Search Tree (BST)? Make a BST for the following sequence of numbers.

45, 36, 76, 23, 89, 115, 98, 39, 41, 56, 69, 48 Traverse the tree in Preorder, Inorder and postorder.

2 What are expression trees? Represent the following expression using a tree. Comment on the result that you get when this tree is traversed in Preorder, Inorder and postorder. $(a-b) / ((c*d)+e)$

3 How do you rotate a Binary Tree? Explain right and left rotations with the help of an example.

4. Which of the following is not the required condition for binary search algorithm?

a. The list must be sorted

b. there should be the direct access to the middle element in any sublist

c. There must be mechanism to delete and/or insert elements in list

d. none of above

5. Which of the following is not a limitation of binary search algorithm?

- a. must use a sorted array
- b. requirement of sorted array is expensive when a lot of insertion and deletions are needed
- c. there must be a mechanism to access middle element directly
- d. binary search algorithm is not efficient when the data elements are more than 1000.

Unit 9 Graphs

Unit summary

Graphs provide the ultimate in data structure flexibility. Graphs can model both real-world systems and abstract problems, so they are used in hundreds of applications.

Here is a small sampling of the range of problems that graphs are routinely applied to.

- Modeling connectivity in computer and communications networks.
- Representing a map as a set of locations with distances between locations; used to compute shortest routes between locations.
- Modeling flow capacities in transportation networks.
- Finding a path from a starting condition to a goal condition; for example, in artificial intelligence problem solving.
- Modeling computer algorithms, showing transitions from one program state to another.
- Finding an acceptable order for finishing subtasks in a complex activity, such as constructing large buildings.
- Modeling relationships such as family trees, business or military organizations, and scientific taxonomies.

General objectives

To make the student familiar with the basic graph terminology and to define the two fundamental representations for graphs, the adjacency matrix and adjacency list. Presentation will be made on graph ADT and simple implementations based on the adjacency matrix and adjacency list. The most commonly used graph traversal algorithms, called depth-first and breadth-first search will be covered. The algorithms for solving some problems related to finding shortest routes in a graph and algorithms for finding the minimum-cost spanning tree, useful for determining lowest-cost connectivity in a network will also be dealt with.

Specific objectives;

Pertinent to this unit can you able to;

- Define the following terms related to graphs:
 - Directed graph
 - Complete graph
 - Undirected graph
 - Weighted graph
 - Vertex Adjacency matrix
 - Edge Adjacency list
 - Path
- Implement a graph using an adjacency matrix to represent the edges
- Explain the difference between a depth-first and a breadth-first search,
- Implement the searching strategies using stacks and queues for auxiliary storage
- Implement a shortest-path operation, using a priority queue to access the edge with the minimum weight
- Identify and use the algorithms for searching and traversing digraphs.

Questions with answers

Q.1 The maximum degree of any vertex in a simple graph with n vertices is

- (A) $n-1$ (B) $n+1$ (C) $2n-1$ (D) n

Ans: A

Q.2 The data structure required for Breadth First Traversal on a graph is

- (A) queue (B) stack (C) array (D) tree

Ans: A

Q.3 A graph with n vertices will definitely have a parallel edge or self loop if the total number of edges are

- (A) greater than $n-1$ (B) less than $n(n-1)$ (C) greater than $n(n-1)/2$ (D) less than $n^2/2$

Ans:A

Q.4 In Breadth First Search of Graph, which of the following data structure is used?

- (A) Stack. (B) Queue. (C) Linked List. (D) None of the above.

Ans. (B)

Q.5 For an undirected graph G with n vertices and e edges, the sum of the degrees of each vertex is

- (A) ne (B) $2n$ (C) $2e$ (D) e^n

Ans. (C)

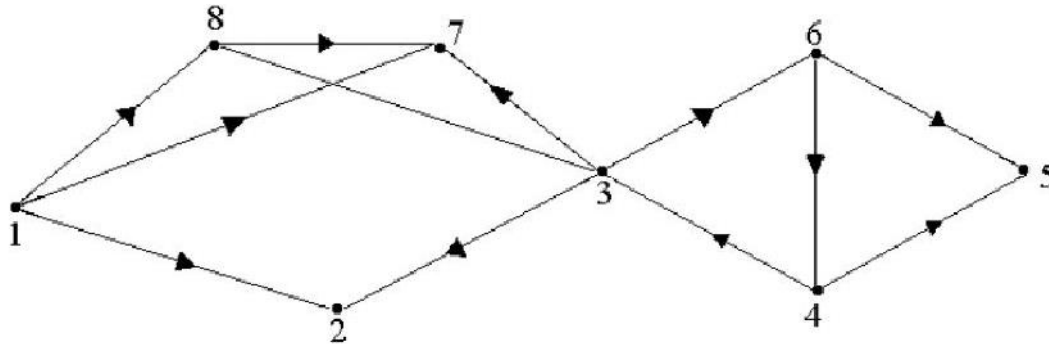
Q.6 The maximum degree of any vertex in a simple graph with n vertices is

- (A) $n-1$ (B) $n+1$ (C) $2n-1$ (D) n

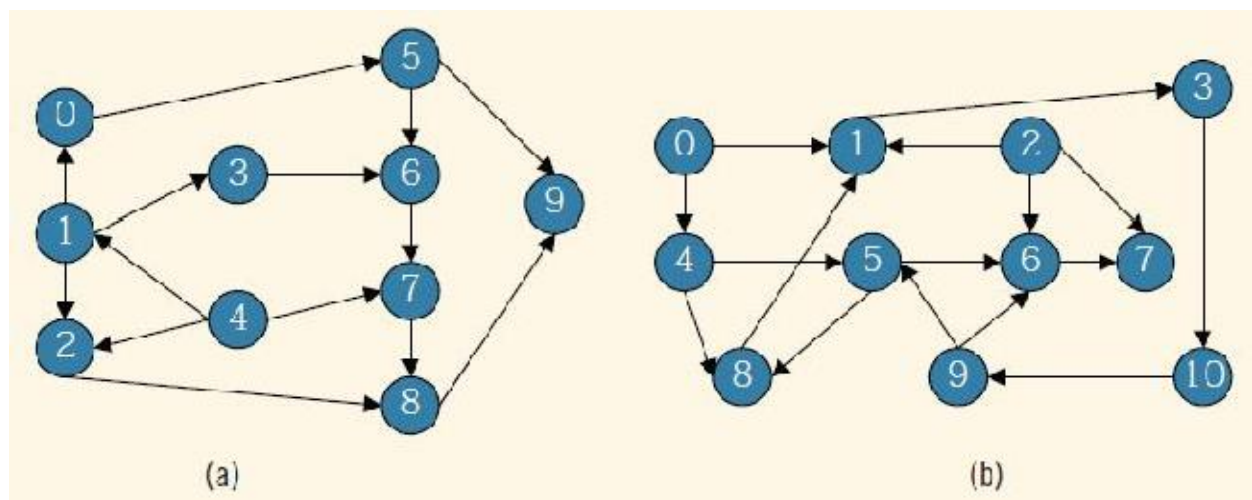
Ans: A

Questions without answers

1. Show the result of running BFS and DFS on the directed graph given below using vertex 3 as source. Show the status of the data structure used at each stage.



2. Which are the two standard ways of traversing a graph? Explain them with an example of each. Use the graph in below for Exercises 1 through 6.



1. In Figure (a), find a path from vertex 0 to vertex 9.
2. In Figure (a), find a path from vertex 0 to vertex 9 via vertex 6.
3. In Figure (a), determine if the graph is simple. Also determine if there is a cycle in this graph.
4. In Figure (a), determine if the vertices 1 and 9 are connected. If these vertices are connected, find a path from vertex 1 to vertex 9.
5. In Figure (b), determine if the vertices 2 and 4 are connected. If these vertices are connected, find a path from vertex 2 to vertex 4.
6. In Figure (b), determine if the graph is simple. Also determine if there is a cycle in this graph.

Study Guide

For

Database Systems Course

Department of Computer Science

Faculty of Informatics



ቅድስት ማርያም ዩኒቨርሲቲ
St. Mary's University, Ethiopia

Prepared by

Worku Alemu

January 2012

Part I Course Title: Fundamentals of Database Systems

Course description

The course covers basic topics related to file handling mechanisms like manual, File-based and database. The various kinds of database architectures, models and issues on database design and implementation will be introduced.

Course objectives

At the end of this course students will be able to:

- Define database terminologies.
- Identify the difference between File-based approaches versus Database approach.
- Know the advantage and disadvantage of database.
- Explain the function of DBMS
- Discuss the three-level database architecture.
- Understand the purpose and importance of conceptual modeling.
- Identify the role of DDL and DML.
- Describe the relational model and properties of database relations.
- Perform conceptual, logical and physical database design.
- Write SQL commands to retrieve, insert, update and delete data and create database objects.

Chapter 1 Introduction to Databases

General objectives

At the end of this chapter students will be able to

- *Discuss the advantages and limitations of the different file handling mechanisms*
- *Explain the components of DBMS*
- *Write the roles of people in database environment*

Specific objectives

At the end of this chapter students will be able to

- Explain some common uses of database systems.
- Discuss the characteristics of file-based systems.
- Identify the problems with the file-based approach.
- Define the meaning of the term 'database'.
- Define the meaning of the term 'database management system' (DBMS).
- Explain the typical functions of a DBMS.
- Identify the major components of the DBMS environment.
- List the personnel involved in the DBMS environment.
- Explain the history of the development of DBMSs.
- Explain the advantages and disadvantages of DBMSs.

Chapter Summary

- The Database Management System (DBMS) is now the underlying framework of the information system and has fundamentally changed the way that many organizations operate. The database system remains a very active research area and many significant problems have still to be satisfactorily resolved.
- The predecessor to the DBMS was the file-based system, which is a collection of application programs that perform services for the end-users, usually the production of reports. Each program defines and manages its own data. Although the file-based system was a great

improvement on the manual filing system, it still has significant problems, mainly the amount of data redundancy present and program–data dependence.

- The database approach emerged to resolve the problems with the file-based approach. A database is a shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization. A DBMS is a software system that enables users to define, create, maintain, and control access to the database. An application program is a computer program that interacts with the database by issuing an appropriate request (typically an SQL statement) to the DBMS. The more inclusive term database system is used to define a collection of application programs that interact with the database along with the DBMS and database itself.
- All access to the database is through the DBMS. The DBMS provides a Data Definition Language (DDL), which allows users to define the database, and a Data Manipulation Language (DML), which allows users to insert, update, delete, and retrieve data from the database.
- The DBMS provides controlled access to the database. It provides security, integrity, concurrency and recovery control, and a user-accessible catalog. It also provides a view mechanism to simplify the data that users have to deal with.
- The DBMS environment consists of hardware (the computer), software (the DBMS, operating system, and applications programs), data, procedures, and people. The people include data and database administrators, database designers, application developers, and end-users.
- Some advantages of the database approach include control of data redundancy, data consistency, sharing of data, and improved security and integrity. Some disadvantages include complexity, cost, reduced performance, and higher impact of a failure.

Review Questions

1.1 Discuss each of the following terms:

- (a) Data
- (b) Database
- (c) Database management system
- (d) Database application program

(e) Data independence

1.2 Describe the approach taken to the handling of data in the early file-based systems. Discuss the disadvantages of this approach.

1.3 Describe the main characteristics of the database approach and contrast it with the file-based approach.

1.4 Describe the five components of the DBMS environment and discuss how they relate to each other.

1.5 Discuss the roles of the following personnel in the database environment:

- (a) Data administrator
- (b) Database administrator
- (c) Logical database designer
- (d) Physical database designer
- (e) Application developer
- (f) end-users.

1.6 Discuss the advantages and disadvantages of DBMSs.

Required Readings

Database System Concepts (Silberschatz 5th Ed): Chapter 1 (1.1 to 1.6)

Database Processing (David M): Chapter 1

Database Systems for Management (James F): Chapter 1

Database Management System (Ramakrishnan): Chapter 1

Fundamental of Relational Database Management System (Sumathi): Chapter 1 (1.1)

Database System Concepts (Silberschatz 5th Ed): Chapter 1 (1.11)

Database Systems for Management (James F): Chapter 2

Fundamental of Relational Database Management System (Sumathi): Chapter 1 (1.12 to 1.14)

Modern Database Management (Jeffrey): Chapter 2

Chapter 2 Database Design

General objectives

At the end of this chapter students will be able to

- *Explain the aims of different phases database design*
- *Explain the different types of database modeling*

- *Write the advantage and limitations of each model*

Specific objectives

In this chapter you will learn:

- The purpose and origin of the three-level database architecture.
- The contents of the external, conceptual, and internal levels.
- The purpose of the external/conceptual and the conceptual/internal mappings.
- The meaning of logical and physical data independence.
- The distinction between a Data Definition Language (DDL) and a Data Manipulation Language (DML).
- A classification of data models.
- The purpose and importance of conceptual modeling.
- The typical functions and services a DBMS should provide.
- The function and importance of the system catalog.
- The software components of a DBMS.
- The meaning of the client–server architecture and the advantages of this type of architecture for a DBMS.
- Discuss types of Database Modeling i.e.
 - ✓ Hierarchical database models
 - ✓ Network database model
 - ✓ Relational model

Chapter Summary

- The ANSI-SPARC database architecture uses three levels of abstraction: external, conceptual, and internal. The external level consists of the users' views of the database. The conceptual level is the community view of the database. It specifies the information content of the entire database, independent of storage considerations.

The conceptual level represents all entities, their attributes, and their relationships, as well as the constraints on the data, and security and integrity information. The internal level is the computer's view of the database. It specifies how data is represented, how records are sequenced, what indexes and pointers exist, and so on.

- The external/conceptual mapping transforms requests and results between the external and conceptual levels. The conceptual/internal mapping transforms requests and results between the conceptual and internal levels.
- A database schema is a description of the database structure. Data independence makes each level immune to changes to lower levels. Logical data independence refers to the immunity of the external schemas to changes in the conceptual schema. Physical data independence refers to the immunity of the conceptual schema to changes in the internal schema.
- A data sublanguage consists of two parts: a Data Definition Language (DDL) and a Data Manipulation Language (DML). The DDL is used to specify the database schema and the DML is used to both read and update the database. The part of a DML that involves data retrieval is called a query language.
- A data model is a collection of concepts that can be used to describe a set of data, the operations to manipulate the data, and a set of integrity constraints for the data. They fall into three broad categories: object-based data models, record-based data models, and physical data models. The first two are used to describe data at the conceptual and external levels; the latter is used to describe data at the internal level.
- Object-based data models include the Entity–Relationship, semantic, functional, and object-oriented models. Record-based data models include the relational, network, and hierarchical models.
- Conceptual modeling is the process of constructing a detailed architecture for a database that is independent of implementation details, such as the target DBMS, application programs, programming languages, or any other physical considerations. The design of the conceptual schema is critical to the overall success of the system. It is worth spending the time and energy necessary to produce the best possible conceptual design.
- Functions and services of a multi-user DBMS include data storage, retrieval, and update; a user-accessible catalog; transaction support; concurrency control and recovery services; authorization services; support for data communication; integrity services; services to promote data independence; utility services.
- The system catalog is one of the fundamental components of a DBMS. It contains ‘data about the data’, or metadata. The catalog should be accessible to users. The Information Resource

Dictionary System is an ISO standard that defines a set of access methods for a data dictionary. This allows dictionaries to be shared and transferred from one system to another.

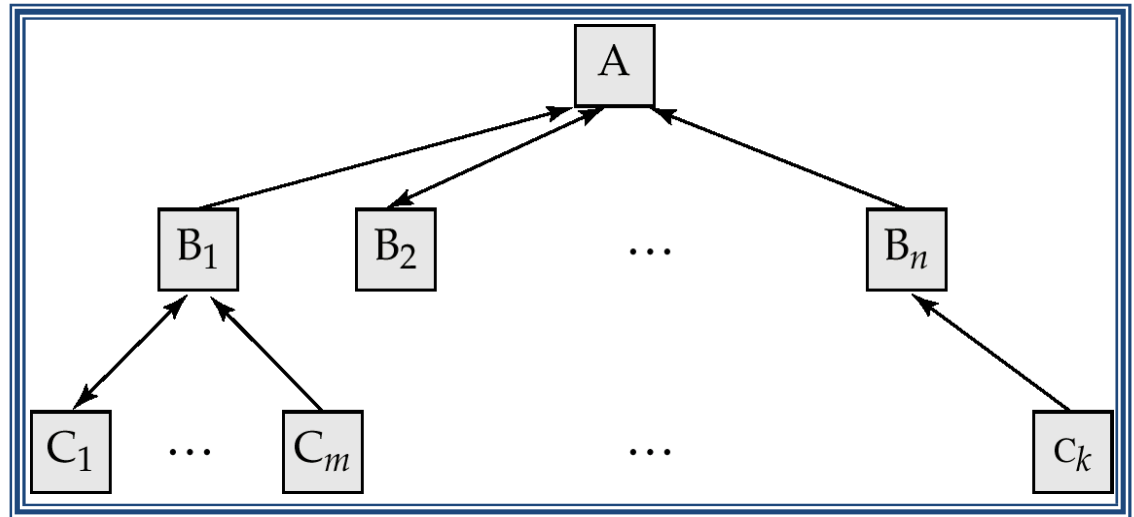
- Client–server architecture refers to the way in which software components interact. There is a client process that requires some resource, and a server that provides the resource. In the two-tier model, the client handles the user interface and business processing logic and the server handles the database functionality. In the Web environment, the traditional two-tier model has been replaced by a three-tier model, consisting of a user interface layer (the client), a business logic and data processing layer (the application server), and a DBMS (the database server), distributed over different machines.
- A data model is a description of the way that data is stored in a database. Data model helps to understand the relationship between entities and to create the most effective structure to hold data.
- The main *purpose* of Data Model is to represent the data in an understandable way. Categories of data models include: Object-based, Record-based, and Physical

Record-based Data Models: Consist of a number of fixed format records. Each record type defines a fixed number of fields; each field is typically of a fixed length. These are Hierarchical Data Model, Network Data Model, Relational Data Model

a. Hierarchical Model

- The simplest data model
- Record type is referred to as node or segment
- The top node is the root node
- Nodes are arranged in a hierarchical structure as sort of upside-down tree
- A parent node can have more than one child node
- A child node can only have one parent node
- The relationship between parent and child is one-to-many
- Relation is established by creating physical link between stored records (each is stored with a predefined access path to other records)
- To add new record type or relationship, the database must be redefined and then stored in a new form.

General Structure



- A parent *may* have an arrow pointing to a child, but a child *must* have an arrow pointing to its parent

b. Network Model

- Allows record types to have more than one parent unlike hierarchical model
- A network data models sees records as set members
- Each set has an owner and one or more members
- Allow no many to many relationship between entities
- Like hierarchical model network model is a collection of physically linked records.
- Allow member records to have more than one owner

c. Relational Data Model

- Relation: Two dimensional table
- Stores information or data in the form of tables (rows and columns)
- A row of the table is called tuple equivalent to record
- A column of a table is called attribute equivalent to fields
- Data value is the value of the Attribute
- Records are related by the data stored jointly in the fields of records in two tables or files. The related tables contain information that creates the relation
- The tables seem to be independent but are related somehow.
- No physical consideration of the storage is required by the user

Review questions

- 2.1 Discuss the concept of data independence and explain its importance in a database environment.
- 2.2 To address the issue of data independence, the ANSI-SPARC three-level architecture was proposed. Compare and contrast the three levels of this model.
- 2.3 What is a data model? Discuss the main types of data model.
- 2.4 Discuss the function and importance of conceptual modeling.
- 2.5 Describe the types of facility you would expect to be provided in a multi-user DBMS.
- 2.6 Discuss the function and importance of the system catalog.
- 2.7 Describe the main components in a DBMS and suggest which components are responsible for each facility.
- 2.8 What is meant by the term 'client–server architecture' and what are the advantages of this approach?
- 2.9 Compare the client–server architecture with two other architectures.

Required Readings

- Database System Concepts (Silberschatz 5th Ed): Chapters 2 (2.1), 7
- Database Processing (David M): Chapters 3,4,5,6
- Database Systems for Management (James F): Chapters 3, 4
- Data Modeling Essentials (Graemec): Chapter 2
- Database Modeling and Design (Resource): Chapters 2,4,5,6
- Fundamental of Relational Database Management System (Sumathi): Chapters 2, 3, 6

Chapter 3. The Relational Model

General objectives: At the end of this chapter students will be able to

- *Explain the aims of selecting relational database model*
- *Describe the different types of relational keys and Relational constraints*
- *Discuss about Relational languages and views*

Specific Objectives

In this chapter you will learn:

- The origins of the relational model.

- The terminology of the relational model.
- How tables are used to represent data.
- The connection between mathematical relations and relations in the relational model.
- Properties of database relations.
- How to identify candidate, primary, alternate, and foreign keys.
- The meaning of entity integrity and referential integrity.
- The purpose and advantages of views in relational systems.

Chapter Summary

- The Relational Database Management System (RDBMS) has become the dominant data-processing software in use today. This software represents the second generation of DBMSs and is based on the relational data model proposed by E. F. Codd.
- A mathematical relation is a subset of the Cartesian product of two or more sets. In database terms, a relation is any subset of the Cartesian product of the domains of the attributes. A relation is normally written as a set of n -tuples, in which each element is chosen from the appropriate domain.
- Relations are physically represented as tables, with the rows corresponding to individual tuples and the columns to attributes.
- Properties of database relations are: each cell contains exactly one atomic value, attribute names are distinct, attribute values come from the same domain, attribute order is immaterial, tuple order is immaterial, and there are no duplicate tuples.
- The degree of a relation is the number of attributes, while the cardinality is the number of tuples. A unary relation has one attribute, a binary relation has two, a ternary relation has three, and an n -ary relation has n attributes.
- A superkey is an attribute, or set of attributes, that identifies tuples of a relation uniquely, while a candidate key is a minimal superkey. A primary key is the candidate key chosen for use in identification of tuples. A relation must always have a primary key. A foreign key is an attribute, or set of attributes, within one relation that is the candidate key of another relation.
- A null represents a value for an attribute that is unknown at the present time or is not applicable for this tuple.

- Entity integrity is a constraint that states that in a base relation no attribute of a primary key can be null. Referential integrity states that foreign key values must match a candidate key value of some tuple in the home relation or be wholly null. Apart from relational integrity, integrity constraints include, required data, domain, and multiplicity constraints; other integrity constraints are called general constraints.
- A view in the relational model is a virtual or derived relation that is dynamically created from the underlying base relation(s) when required. Views provide security and allow the designer to customize a user's model. Not all views are updatable.

Review Questions

3.1 Discuss each of the following concepts in the context of the relational data model:

(a) Relation, (b) Attribute, (c) Domain (d) Tuple, (e) Degree and cardinality.

3.2 Describe the relationship between mathematical relations and relations in the relational data model.

3.3 Discuss the properties of a relation.

3.4 Discuss the differences between the candidate keys and the primary key of a relation. Explain what is meant by a foreign key. How do foreign keys of relations relate to candidate keys? Give examples to illustrate your answer.

3.5 Define the two principal integrity rules for the relational model. Discuss why it is desirable to these rules.

3.6 What is a view? Discuss the difference between a view and a base relation.

Required Readings

Database System Concepts (Silberschatz 5th Ed): Chapter 2 (2.2 to 2.3)

Database Systems for Management (James F): Chapter 6

Database Management System (Ramakrishnan): Chapter 4

Chapter 4 Relational Algebra

General objectives

At the end of this chapter students will be able to

- Identify the different relational operators
- Write the results of different operations

- Write Queries in Relational Algebra

Specific Objectives

At the end of this chapter students will be able to

- Examine the two unary operations Selection and Projection.
- Examine the binary operations of the relational algebra, starting with the set operations of Union, Set difference, Intersection, and Cartesian product.
- Use various forms of Join operation, Theta join, Equijoin, Natural join, Outer join, Semi-join to write queries and write the results

Chapter Summary

- The relational algebra is a (high-level) procedural language: it can be used to tell the DBMS how to build a new relation from one or more relations in the database. The relational calculus is a non-procedural language: it can be used to formulate the definition of a relation in terms of one or more database relations. However, formally the relational algebra and relational calculus are equivalent to one another: for every expression in the algebra, there is an equivalent expression in the calculus (and vice versa).
- The relational calculus is used to measure the selective power of relational languages. A language that can be used to produce any relation that can be derived using the relational calculus is said to be relationally complete. Most relational query languages are relationally complete but have more expressive power than the relational algebra or relational calculus because of additional operations such as calculated, summary, and ordering functions.
- The five fundamental operations in relational algebra, *Selection*, *Projection*, *Cartesian product*, *Union*, and *Set difference*, perform most of the data retrieval operations that we are interested in. In addition, there are also the *Join*, *Intersection*, and *Division* operations, which can be expressed in terms of the five basic operations.

Review questions

- 4.1. Explain the term closure of relational operations.
- 4.2. Define the five basic relational algebra operations.
- 4.3. Discuss the differences between the five Join operations: Theta join, Equijoin, Natural join, Outer join, and Semi join. Give examples to illustrate your answer.

Required Readings

Database System Concepts (Silberschatz 5th Ed): Chapter 2

Fundamental of Relational Database Management System (Sumathi): Chapters 5

Chapter 5 SQL

General objectives: *At the end of this chapter students will be able to*

- *Explain the aims and history and importance of SQL*
- *Explain the different types of SQL commands*

Specific objectives: *At the end of this chapter students will be able to*

- *Write the different SQL commands to create tables, store and retrieve data*
- *Explain the objectives of creating views*
- *Create and remove views*
- *Explain the different Restrictions on views*
- *Write the different SQL commands to create tables, store and retrieve data*

Chapter Summary

- SQL is a non-procedural language, consisting of standard English words such as SELECT, INSERT,DELETE, that can be used by professionals and non-professionals alike. It is both the formal and *de facto* standard language for defining and manipulating relational databases.
- The SELECT statement is the most important statement in the language and is used to express a query. It combines the three fundamental relational algebra operations of *Selection*, *Projection*, and *Join*. Every SELECT statement produces a query result table consisting of one or more columns and zero or more rows.
- The SELECT clause identifies the columns and/or calculated data to appear in the result table. All column names that appear in the SELECT clause must have their corresponding tables or views listed in the FROM clause.
- The WHERE clause selects rows to be included in the result table by applying a search condition to the rows of the named table(s). The ORDER BY clause allows the result table to be sorted on the values in one or more columns. Each column can be sorted in ascending or descending order. If specified, the ORDER BY clause must be the last clause in the SELECT statement.

- SQL supports five aggregate functions (COUNT, SUM, AVG, MIN, and MAX) that take an entire column as an argument and compute a single value as the result. It is illegal to mix aggregate functions with column names in a SELECT clause, unless the GROUP BY clause is used.
- The GROUP BY clause allows summary information to be included in the result table. Rows that have the same value for one or more columns can be grouped together and treated as a unit for using the aggregate functions. In this case the aggregate functions take each group as an argument and compute a single value for each group as the result. The HAVING clause acts as a WHERE clause for groups, restricting the groups that appear in the final result table. However, unlike the WHERE clause, the HAVING clause can include aggregate functions.
- A subselect is a complete SELECT statement embedded in another query. A subselect may appear within the WHERE or HAVING clauses of an outer SELECT statement, where it is called a subquery or nested query. Conceptually, a subquery produces a temporary table whose contents can be accessed by the outer query. A subquery can be embedded in another subquery.
- There are three types of subquery: scalar, row, and table. A *scalar subquery* returns a single column and a single row; that is, a single value. In principle, a scalar subquery can be used whenever a single value is needed. A *row subquery* returns multiple columns, but again only a single row. A row subquery can be used whenever a row value constructor is needed, typically in predicates. A *table subquery* returns one or more columns and multiple rows. A table subquery can be used whenever a table is needed, for example, as an operand for the IN predicate. If the columns of the result table come from more than one table, a join must be used, by specifying more than one table in the FROM clause and typically including a WHERE clause to specify the join column(s). The ISO standard allows Outer joins to be defined. It also allows the set operations of *Union*, *Intersection*, and *Difference* to be used with the UNION, INTERSECT, and EXCEPT commands.
- As well as SELECT, the SQL DML includes the INSERT statement to insert a single row of data into a named table or to insert an arbitrary number of rows from one or more other tables using a sub-select; the UPDATE statement to update one or more values in a specified column or columns of a named table; the DELETE statement to delete one or more rows from a named table.

Review Questions

- 5.1 What are the two major components of SQL and what function do they serve?
- 5.2 What are the advantages and disadvantages of SQL?
- 5.3 Explain the function of each of the clauses in the SELECT statement. What restrictions are imposed on these clauses?
- 5.4 What restrictions apply to the use of the aggregate functions within the SELECT statement? How do nulls affect the aggregate functions?
- 5.5 Explain how the GROUP BY clause works. What is the difference between the WHERE and HAVING clauses?
- 5.6 What is the difference between a subquery and a join? Under what circumstances would you not be able to use a subquery?

Recommended readings

- Database System Concepts (Silberschatz 5th Ed): Chapter 3
- Database Processing (David M): Chapter 7
- Database Systems for Management (James F): Chapter 5
- Database Management System (Ramakrishnan): Chapter 1
- Fundamental of Relational Database Management System (Sumathi): Chapter 4
- Modern Database management (Jeffery): Chapter 7
- Database System Concepts (Silberschatz 5th Ed): Chapter 4
- Database Management System (Ramakrishnan): Chapter 5 (5.7)
- Modern Database management (Jeffery): Chapter 8
- Fundamental of Relational Database Management System (Sumathi): Chapter 4 (4.15)

Chapter 6 Integrity and Security

General objectives: At the end of this chapter students will be able to

- *Identify Integrity constraints that ensure that changes made to the database by authorized users do not result in a loss of data consistency.*
- *Explain integrity constraints guard against accidental damage to the database.*

Specific objectives: At the end of this chapter students will be able to

- *Identify the different types of entity integrities*

- *Explain the roles of integrity rules*
- *Apply integrity rules on relational database*
- *Identify the different types of database security mechanisms*
- *Explain data protection and Privacy laws*

Chapter Summary

- Database security is the mechanisms that protect the database against intentional or accidental threats.
- Database security is concerned with avoiding the following situations: theft and fraud, loss of confidentiality (secrecy), loss of privacy, loss of integrity, and loss of availability.
- A threat is any situation or event, whether intentional or accidental, that will adversely affect a system and consequently an organization.
- Computer-based security controls for the multi-user environment include: authorization, access controls, views, backup and recovery, integrity, encryption, and RAID technology.
- Authorization is the granting of a right or privilege that enables a subject to have legitimate access to a system or a system's object. Authentication is a mechanism that determines whether a user is who he or she claims to be.
- Most commercial DBMSs provide an approach called Discretionary Access Control (DAC), which manages privileges using SQL. The SQL standard supports DAC through the GRANT and REVOKE commands. Some commercial DBMSs also provide an approach to access control called Mandatory Access Control (MAC), which is based on system-wide policies that cannot be changed by individual users. In this approach each database object is assigned a *security class* and each user is assigned a *clearance* for a security class, and *rules* are imposed on reading and writing of database objects by users. The SQL standard does not include support for MAC.
- A view is the dynamic result of one or more relational operations operating on the base relations to produce another relation. A view is a virtual relation that does not actually exist in the database but is produced upon request by a particular user at the time of request. The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users.

- Backup is the process of periodically taking a copy of the database and log file (and possibly programs) on to offline storage media. Journaling is the process of keeping and maintaining a log file (or journal) of all changes made to the database to enable recovery to be undertaken effectively in the event of a failure.
- Integrity constraints also contribute to maintaining a secure database system by preventing data from becoming invalid, and hence giving misleading or incorrect results.
- Encryption is the encoding of the data by a special algorithm that renders the data unreadable by any program without the decryption key.

Review Questions

6.1 Explain the purpose and scope of database security.

6.2 List the main types of threat that could affect a database system and for each describe the controls that you would use to counteract each of them.

6.3 Explain the following in terms of providing security for a database:

- (a) Authorization; (b). Access controls; (c) . Views; (d). Backup and recovery; (e). integrity;
- (f) Encryption;

Chapter 7 Transaction Management

General objectives: At the end of this chapter students will be able to

- *Identify the different types of recovery control*

Specific objectives: At the end of this chapter students will be able to

- *Explain the need for Recovery Control*
- *Explain the need for concurrency Control*

Chapter Summary

- Concurrency control is the process of managing simultaneous operations on the database without having them interfere with one another. Database recovery is the process of restoring the database to a correct state after a failure. Both protect the database from inconsistencies and data loss.

A transaction is an action, or series of actions, carried out by a single user or application program, which accesses or changes the contents of the database. A transaction is a logical *unit*

of work that takes the database from one consistent state to another. Transactions can terminate successfully (commit) or unsuccessfully (abort). Aborted transactions must be undone or rolled back. The transaction is also the *unit of concurrency* and the *unit of recovery*.

- A transaction should possess the four basic or so-called ACID, properties: atomicity, consistency, isolation, and durability. Atomicity and durability are the responsibility of the recovery subsystem; isolation and, to some extent, consistency are the responsibility of the concurrency control subsystem.

Concurrency control is needed when multiple users are allowed to access the database simultaneously. Without it, problems of *lost update*, *uncommitted dependency*, and *inconsistent analysis* can arise.

- Serial execution means executing one transaction at a time, with no interleaving of operations. A schedule shows the sequence of the operations of transactions. A schedule is serializable if it produces the same results as some serial schedule.
- Two methods that guarantee serializability are two-phase locking (2PL) and time stamping. Locks may be shared (read) or exclusive (write). In two-phase locking, a transaction acquires all its locks before releasing any. With timestamping, transactions are ordered in such a way that older transactions get priority in the event of conflict.
- Deadlock occurs when two or more transactions are waiting to access data the other transaction has locked. The only way to break deadlock once it has occurred is to abort one or more of the transactions.
- A tree may be used to represent the granularity of locks in a system that allows locking of data items of different sizes. When an item is locked, all its descendants are also locked. When a new transaction requests a lock, it is easy to check all the ancestors of the object to determine whether they are already locked. To show whether any of the node's descendants are locked, an intention lock is placed on all the ancestors of any node being locked.
- Some causes of failure are system crashes, media failures, application software errors, carelessness, natural physical disasters, and sabotage. These failures can result in the loss of main memory and/or the disk copy of the database. Recovery techniques minimize these effects.
- To facilitate recovery, one method is for the system to maintain a log file containing transaction records that identify the start/end of transactions and the before- and after-images of the write

operations. Using deferred updates, writes are done initially to the log only and the log records are used to perform actual updates to the database. If the system fails, it examines the log to determine which transactions it needs to redo, but there is no need to undo any writes. Using immediate updates, an update may be made to the database itself any time after a log record is written. The log can be used to undo and redo transactions in the event of failure.

- Checkpoints are used to improve database recovery. At a checkpoint, all modified buffer blocks, all log records, and a checkpoint record identifying all active transactions are written to disk. If a failure occurs, the checkpoint record identifies which transactions need to be redone.
- Advanced transaction models include nested transactions, sagas, multilevel transactions, dynamically restructuring transactions, and workflow models.

Review Questions

- 7.1 Explain what is meant by a transaction.
- 7.2 The consistency and reliability aspects of transactions are due to the 'ACIDity' properties of transactions. Discuss each of these properties and how they relate to the concurrency control and recovery mechanisms. Give examples to illustrate your answer.
- 7.3 Describe, with examples, the types of problem that can occur in a multi-user environment when concurrent access to the database is allowed.
- 7.4. Discuss how the concurrency control mechanism interacts with the transaction mechanism.
- 7.5 Explain the concepts of serial, non-serial, and serializable schedules. State the rules for equivalence of schedules.
- 7.6 Discuss the difference between conflict serializability and view serializability.
- 7.7 Discuss the types of problem that can occur with locking-based mechanisms for concurrency control and the actions that can be taken by a DBMS to prevent them.
- 7.8 Why would two-phase locking not be an appropriate concurrency control scheme for indexes?
- 7.9 What is a timestamp? How do time stamp based protocols for concurrency control differ from locking based protocols?
- 7.10 Describe the basic timestamp ordering protocol for concurrency control. What is Thomas's write rule and how does this affect the basic timestamp ordering protocol?
- 7.11 Describe how versions can be used to increase concurrency.
- 7.12 Discuss the difference between pessimistic and optimistic concurrency control.

7.13 Discuss the types of failure that may occur in a database environment. Explain why it is important for a multi-user DBMS to provide a recovery mechanism.

7.14 Discuss how the log file (or journal) is a fundamental feature in any recovery mechanism.

Explain what is meant by forward and backward recovery and describe how the log file is used in forward and backward recovery. What is the significance of the write-ahead log protocol?

How do checkpoints affect the recovery protocol?

7.15 Compare and contrast the deferred update and immediate update recovery protocols.

Required Readings

Database System Concepts (Silberschatz 5th Ed): Chapters 15, 16

Database Management System (Ramakrishnan): Chapters 16, 17

Fundamental of Relational Database Management System (Sumathi): Chapter 7

Part II Course Title: Advanced Database Systems

Chapter 8: Relational Database Design

Enhanced Entity–Relationship Modeling

General objectives:

At the end of this unit you will be able to: Explain the limitations of the basic concepts of the Entity–Relationship (ER) model and the requirements to represent more complex applications using additional data modeling concepts.

Specific objectives

At the end of this unit you will be able to:

- Identify the most useful additional data modeling concepts of the Enhanced Entity–Relationship (EER) model called specialization/generalization, aggregation, and composition.
- Identify a diagrammatic technique for displaying specialization/generalization, aggregation, and composition in an EER diagram using the Unified Modeling Language (UML).

Chapter Summary

- A super class is an entity type that includes one or more distinct sub groupings of its occurrences, which require to be represented in a data model. A subclass is a distinct sub grouping of occurrences of an entity type, which require to be represented in a data model.
- Specialization is the process of maximizing the differences between members of an entity by identifying their distinguishing features.
- Generalization is the process of minimizing the differences between entities by identifying their common features.
- There are two constraints that may apply to a specialization/generalization called participation constraints and disjoint constraints.
- A participation constraint determines whether every member in the super class must participate as a member of a subclass.
- A disjoint constraint describes the relationship between members of the subclasses and indicates whether it is possible for a member of a super class to be a member of one, or more than one, subclass.
- Aggregation represents a 'has-a' or 'is-part-of' relationship between entity types, where one represents the 'whole' and the other the 'part'.
- Composition is a specific form of aggregation that represents an association between entities, where there is a strong ownership and coincidental lifetime between the 'whole' and the 'part'.

Review Questions

- 8.1 Describe what a superclass and a subclass represent.
- 8.2 Describe the relationship between a super class and its subclass.
- 8.3 Describe and illustrate using an example the process of attribute inheritance.
- 8.4 What are the main reasons for introducing the concepts of super classes and subclasses into an ER model?
- 8.5 Describe what a shared subclass represents and how this concept relates to multiple inheritances.
- 8.6 Describe and contrast the process of specialization with the process of generalization.

8.7 Describe the two main constraints that apply to a specialization/generalization relationship.

8.8 Describe and contrast the concepts of aggregation and composition and provide an example of each.

Required Readings

Database Systems Thomas Connolly Carolyn Besg Fourth edition Chapter 12

Chapter 9: Database implementation and Tools

General objective:

At the end of this chapter you will be able to : *Explain Data Design and Implementation in an Organization*

Specific objectives:

At the end of this unit you will be able to:

- Describe the use of UML and its support for database design specifications
- Describe representing specialization and generalization in UML Class diagram.
- Describe UML based design tools
- Identify automated database design tools.

Chapter Summary

- Database design mainly involves the design of the database schema. The entity relationship (E-R) data model is a widely used data model for database design. It provides a convenient graphical representation to view data, relationships, and constraints.
- The model is intended primarily for the database-design process. It was developed to facilitate database design by allowing the specification of an enterprise schema. Such a schema represents the overall logical structure of the database. This overall structure can be expressed graphically by an E-R diagram.
- An entity is an object that exists in the real world and is distinguishable from other objects. We express the distinction by associating with each entity a set of attributes that describes the object.

- A relationship is an association among several entities. A relationship set is a collection of relationships of the same type, and an entity set is a collection of entities of the same type.
- A superkey of an entity set is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set. We choose a minimal superkey for each entity set from among its superkeys; the minimal superkey is termed the entity set's primary key. Similarly, a relationship set is a set of one or more attributes that, taken collectively, allows us to identify uniquely a relationship in the relationship set. Likewise, we choose a minimal superkey for each relationship set from among its superkeys; this is the relationship set's primary key.
- Mapping cardinalities express the number of entities to which another entity can be associated via a relationship set.
- An entity set that does not is termed a weak entity set, strong entity set.
- Specialization and generalization define a containment relationship between a higher-Level entity set and one or more lower-level entity sets. Specialization is the result of taking a subset of a higher-level entity set to form a lower level entity set. Generalization is the result of taking the union of two or more have sufficient attributes to form a primary key
- Aggregation is an abstraction in which relationship sets (along with their associated entity sets) are treated as higher-level entity sets, and can participate in relationships.
- The various features of the E-R model offer the database designer numerous choices in how to best represent the enterprise being modeled. Concepts and objects may, in certain cases, be represented by entities, relationships, or attributes.
- A database design specified by an E,R diagram can be represented by a collection of relation schemas. For each entity set and for each relationship set in the database there is a unique relation schema hat is assigned the name of the corresponding entity set or relationship set. This forms the basis for deriving a relational database design from an E-R diagram.
- The Unified Modeling Language (UML) provides a graphical means of modeling various components of a software system. The class diagram component of UML is based on E-R diagrams. However, there are some differences between the two that one must beware of.

Review questions

9.1 Explain the distinctions among the terms primary key, candidate key, and superkey.

9.2 Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors.

Associate with each patient a log of the various tests and examinations conducted.

9.3 Explain the difference between a weak and a strong entity set.

9.4 Define the concept of aggregation. Give two examples of where this concept is useful.

9.5 Design a generalization-specialization hierarchy for a motor vehicle sales company. The company sells motorcycles, passenger cars, vans, and buses. Justify your placement of attributes at each level of the hierarchy. Explain why they should not be placed at a higher or lower level.

9.6 Explain the distinction between total and partial constraints

Chapter 10: Advanced SQL in Oracle

General objectives:

At the end of this chapter you will be able to: Discuss Features and basic architecture, Database Design and Querying Tools, SQL Variations and extensions

Specific objectives: **At the end of this chapter you will be able to**

- Explain Storage and Indexing, Query Processing, evaluation and Optimization, Assertion and views, Cursors, triggers and stored procedures
 - Identify Embedded SQL, dynamic SQL, and Advanced Features of SQL, System Catalog in Oracle
- Identify SQL Variations and Extensions, Transaction Management, Storage and Indexing ,Query Processing and evaluation and optimization

Chapter Summary

Database Structure and Space Management Overview

- An Oracle **database** is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a **server** reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery

Database Design and Querying Tools

- Oracle provides a variety of tools for database design, querying, report generation, and data analysis, including OLAP.

Database Design Tools

Most of Oracle's design tools are included in the Oracle Developer Suite. This is a suite of tools for various aspects of application development, including tools for forms development, data modeling, reporting, and querying. The suite supports the UML standard for development modeling.

- The major database design tool in the suite is Oracle Designer, which translates business logic and data flows into schema definitions and procedural scripts for application logic. It supports such modeling techniques as E-R diagrams, information engineering, and object analysis and design. Oracle Designer stores the design in Oracle Repository, which serves as a single point of metadata for the application.

Querying Tools

- Oracle provides tools for ad-hoc querying, report generation and data analysis, including OLAP.

Oracle Application Server Discoverer is a Web-based, ad-hoc query, reporting, analysis, and Web publishing tool for end-users and data analysts. It allows users to drill up and down on result sets, pivot data, and store calculations as reports that can be published in a variety of formats such as spreadsheets or HTML.

Variations and Extensions

Oracle supports all core SQL: 1999 features fully or partially, with some minor exceptions such as distinct data types. In addition, Oracle supports a large number of other language constructs, some of which conform with SQL:1999, while others are Oracle-specific in syntax or functionality.

Object-Relational Features

Oracle has extensive support for object-relational constructs, including:

- Object types single-inheritance model is supported for type hierarchies.
- Collection types. Oracle supports arrays, which are variable-length arrays, and nested tables.
- Object tables. These are used to store objects while providing a relational view of the attributes of the objects.
- Object views. These provide a virtual object table view of data stored in a regular relational table. They allow data to be accessed or viewed in an object oriented style even if the data are really stored in a traditional relational format.

- User-defined aggregate functions. These can be used in SQL statements in the same way as built-in functions such as sum and count.

Triggers

- Oracle provides several types of triggers and several options for when and how they are invoked. For triggers that execute on DML statements such as insert, update, and delete,
- Oracle supports row triggers and statement triggers. Row triggers execute once for every row that is affected (updated or deleted, for example) by the DML operation.
- Oracle also has triggers that execute on a variety of other events, like database start-up or shut down/ server error messages/ user logon or logoff, and DDL statements such as create, alter, and drop statements.

Storage and Indexing

- In Oracle parlance, a database consists of information stored in files and is accessed through an instance, which is a shared-memory area and a set of processes that interact with the data in the files.

Tables

- A standard table in Oracle is heap organized; that is, the storage location of a row in a table is not based on the values contained in the row, and is fixed when the row is inserted.
- Oracle supports nested tables; that is, a table can have a column whose data type is another table. The nested table is not stored in line in the parent table, but is stored in a separate table.
- Oracle supports temporary tables where the duration of the data is either the transaction in which the data are inserted or the user session. The data are private to the session and are automatically removed at the end of its duration.

Indices

Oracle supports several different types of indices. The most commonly used type is what Oracle (and several other vendors) call a B-tree index created on one or multiple columns.

Partitioning

Oracle supports various kinds of horizontal partitioning of tables and indices, and this feature plays a major role in Oracle's ability to support very large databases. The ability to partition a table or index has advantages in many areas

- Backup and recovery are easier and faster, since they can be done on individual partitions rather than on the table as a whole.
- Loading operations in a data warehousing environment are less intrusive:

Data can be added to a partition, and then the partition added to a table, which is an instantaneous operation. Likewise, dropping a partition with obsolete data from a table is very easy in a data warehouse that maintains a rolling window of historical data.

Query Processing and Optimization

- Oracle supports a large variety of processing techniques in its query-processing engine.

Some of the more important ones are described here briefly.

Execution Methods

Data can be accessed through a variety of access methods:

- Full table scan. The query processor scans the entire table by getting information about the blocks that make up the table from the extent map and scanning those blocks.
- Index fast full scan. The processor scans the extents in the same way as the table extent in a full table scan.

Parallel Execution

- Oracle allows the execution of a single SQL statement to be parallelized by dividing the work between multiple processes on a multiprocessor computer. This feature is especially useful for computationally intensive operations that would otherwise take an unacceptably long time to perform.

Concurrency control and Recovery

- Oracle supports concurrency-control and recovery techniques that provide a number of useful features.

Concurrency Control

Oracle's multi version concurrency control differs from the concurrency mechanisms used by most other database vendors. Read-only queries are given a read-consistent snapshot, which is a view of the

database as it existed at a specific point in time, containing all updates that were committed by that point in time, and not containing any updates that were not committed at that point in time. Thus, read locks are not used and read-only queries do not interfere with other database activity in terms of locking.

Database Administration Tools

- Oracle provides users a number of tools and features for system management and application development. In the release Oracle10g, much emphasis was put on the concept of manageability that is, reducing the complexity of all aspects of creating and administering an Oracle database. This effort covered a wide variety of areas, including database creation, tuning, space management, storage management, backup and recovery, memory management, performance diagnostics, and workload management.

Database Resource Management

A database administrator needs to be able to control how the processing power of the hardware is divided among users all groups of users. Some groups may execute interactive queries where response time is critical; others may execute long-running reports that can be run as batch jobs in the background when the system load is low.

It is also important to be able to prevent a user from inadvertently submitting an extremely expensive ad-hoc query that will unduly delay other users.

Chapter 11: Query Processing and Evaluation

General objectives

At the end of this chapter you will be able to: Identify the different Measures of Query Cost

Specific objectives

At the end of this chapter you will be able to:

- Explain the Role of Relational Algebra and Relational Calculus in query optimization
- Explain Estimating Statistics of Expression

- Choice of Evaluation Plans
- Views and query processing
- Storage and query optimization

Chapter Summary

- The aims of **query processing** are to transform a query written in a high-level language, typically SQL, into a correct and efficient execution strategy expressed in a low-level language like the relational algebra, and to execute the strategy to retrieve the required data.
- As there are many equivalent transformations of the same high-level query, the DBMS has to choose the one that minimizes resource usage. This is the aim of **query optimization**. Since the problem is computationally intractable with a large number of relations, the strategy adopted is generally reduced to finding a near-optimum solution.
- There are two main techniques for query optimization, although the two strategies are usually combined in practice. The first technique uses **heuristic rules** that order the operations in a query. The other technique compares different strategies based on their relative costs, and selects the one that minimizes resource usage.
- Query processing can be divided into four main phases: decomposition (consisting of parsing and validation), optimization, code generation, and execution. The first three can be done either at compile time or at runtime.
- **Query decomposition** transforms a high-level query into a relational algebra query, and checks that the query is syntactically and semantically correct. The typical stages of query decomposition are analysis, normalization, semantic analysis, simplification, and query restructuring. A **relational algebra tree** can be used to provide an internal representation of a transformed query.
- **Query optimization** can apply transformation rules to convert one relational algebra expression into an equivalent expression that is known to be more efficient. Transformation rules include cascade of selection, commutativity of unary operations, commutativity of Theta join (and

Cartesian product), commutativity of unary operations and Theta join (and Cartesian product), and associativity of Theta join (and Cartesian product).

- **Heuristics rules** include performing Selection and Projection operations as early as possible; combining Cartesian product with a subsequent Selection whose predicate represents a join condition into a Join operation; using associativity of binary operations to rearrange leaf nodes so that leaf nodes with the most restrictive Selections are executed first.
- **Cost estimation** depends on statistical information held in the system catalog. Typical statistics include the cardinality of each base relation, the number of blocks required to store a relation, the number of distinct values for each attribute, the selection cardinality of each attribute, and the number of levels in each multilevel index.

Review Questions

- 11.1 What are the objectives of query processing?
- 11.2 How does query processing in relational systems differ from the processing of low-level query languages for network and hierarchical systems?
- 11.3 What are the typical phases of query processing?
- 11.4 What are the typical stages of query decomposition?
- 11.5 What is the difference between conjunctive and disjunctive normal form?
- 11.6 How would you check the semantic correctness of a query?
- 11.7 State the transformation rules that apply to:
(a) Selection operations, (b) Projection operations (c) Theta join operations.
- 11.8 State the heuristics that should be applied to improve the processing of a query.
- 11.9 What types of statistics should a DBMS hold to be able to derive estimates of relational algebra operations?
- 11.10 Under what circumstances would the system have to resort to a linear search when implementing a Selection operation?

Chapter 12: Distributed Databases

General objectives

At the end of this chapter you will be able to: Explain the need for distributed databases.

Specific objectives:

At the end of this chapter you will be able to:

- The differences between distributed database systems, distributed processing, and parallel database systems.
- Explain the advantages and disadvantages of distributed DBMSs.
- Identify the problems of heterogeneity in a distributed DBMS.
- Explain the functions that should be provided by a distributed DBMS.
- Explain architecture for a distributed DBMS.
- Identify the main issues associated with distributed database design, namely fragmentation, replication, and allocation.
- Explain how fragmentation should be carried out.
- The importance of allocation and replication in distributed databases.

Chapter Summary

- A distributed database is a logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network. The DDBMS is the software that transparently manages the distributed database.
- A DDBMS is distinct from distributed processing, where a centralized DBMS is accessed over a network. It is also distinct from a parallel DBMS, which is a DBMS running across multiple processors and disks and which has been designed to evaluate operations in parallel, whenever possible, in order to improve performance.
- The advantages of a DDBMS are that it reflects the organizational structure; it makes remote data more shareable, it improves reliability, availability, and performance; it may be more economical, it provides for modular growth, facilitates integration, and helps organizations remain competitive. The major disadvantages are cost, complexity, lack of standards, and experience.
- A DDBMS may be classified as homogeneous or heterogeneous. In a homogeneous system, all sites use the same DBMS product. In a heterogeneous system, sites may run different DBMS products, which need not be based on the same underlying data model, and so the system may be composed of relational, network, hierarchical, and object-oriented DBMSs.

- A multi database system (MDBS) is a distributed DBMS in which each site maintains complete autonomy. An MDBS resides transparently on top of existing database and file systems, and presents a single database to its users. It maintains a global schema against which users issue queries and updates; an MDBS maintains only the global schema and the local DBMSs themselves maintain all user data.
- Communication takes place over a network, which may be a local area network (LAN) or a wide area network (WAN). LANs are intended for short distances and provide faster communication than WANs. A special case of the WAN is a metropolitan area network (MAN), which generally covers a city or suburb.
- As well as having the standard functionality expected of a centralized DBMS, a DDBMS will need extended communication services, extended system catalog, distributed query processing, and extended security, concurrency, and recovery services.
- A relation may be divided into a number of sub relations called fragments, which are allocated to one or more sites. Fragments may be replicated to provide improved availability and performance.
- There are two main types of fragmentation: horizontal and vertical. Horizontal fragments are subsets of tuples and vertical fragments are subsets of attributes. There are also two other types of fragmentation: mixed and derived a type of horizontal fragmentation where the fragmentation of one relation is based on the fragmentation of another relation.
- The definition and allocation of fragments are carried out strategically to achieve locality of reference, improved reliability and availability, acceptable performance, balanced storage capacities and costs, and minimal communication costs. The three correctness rules of fragmentation are: completeness, reconstruction, and disjointness.
- There are four allocation strategies regarding the placement of data: centralized (a single centralized database), fragmented (fragments assigned to one site), complete replication (complete copy of the database maintained at each site), and selective replication (combination of the first three).
- The DDBMS should appear like a centralized DBMS by providing a series of transparencies. With distribution transparency, users should not know that the data has been fragmented/replicated. With transaction transparency, the consistency of the global database should be

maintained when multiple users are accessing the database concurrently and when failures occur. With performance transparency, the system should be able to efficiently handle queries that reference data at more than one site. With DBMS transparency, it should be possible to have different DBMSs in the system.

Review question

- 12.1 Explain what is meant by a DDBMS and discuss the motivation in providing such a system.
- 12.2 Compare and contrast a DDBMS with distributed processing. Under what circumstances would you choose a DDBMS over distributed processing?
- 12.3 Compare and contrast a DDBMS with a parallel DBMS. Under what circumstances would you choose a DDBMS over a parallel DBMS?
- 12.4 Discuss the advantages and disadvantages of a DDBMS.
- 12.5 What is the difference between a homogeneous and a heterogeneous DDBMS? Under what circumstances would such systems generally arise?
- 12.6 What are the main differences between LAN and WAN?
- 12.7 What functionality do you expect in a DDBMS?
- 12.8 What is a multi-database system? Describe reference architecture for such a system.

Chapter 13: Object Oriented Database

General objectives:

In this chapter you will be able to: Explain the limitations of Relational databases and the need of Object oriented databases

Specific objectives:

- Identify Complex Data Types
- Explain Structured Types and Inheritance in SQL
- Data types (arrays, multi-set etc) and structure in Object oriented databases using SQL
- Object-Identity and Reference Types in SQL
- Explain ODL and OQL
- Compare Object-Oriented versus Object-Relational

- Give examples of Object oriented and object relational database implementation

Chapter Summery

- The object-relational data model extends the relational data model by providing a richer type system including collection types and object orientation.
- Collection tuples include nested relations, sets, multi-sets, and arrays, and the object-relational model permits attributes of a table to be collections.
- Object orientation provides inheritance with subtypes and sub-tables, as well as object (tuple) references
- Object-relational database systems (that is, database systems based on the object-relation model) provide a convenient migration path for users of relational databases who wish to use object-oriented features
- The relational model, and relational systems in particular, have weaknesses such as poor representation of 'real world' entities, semantic overloading, poor support for integrity and enterprise constraints, limited operations, and impedance mismatch. The limited modeling capabilities of relational DBMSs have made them unsuitable for advanced database applications.
- The concept of encapsulation means that an object contains both a data structure and the set of operations that can be used to manipulate it. The concept of information hiding means that the external aspects of an object are separated from its internal details, which are hidden from the outside world.
- An object is a uniquely identifiable entity that contains both the attributes that describe the state of a 'real world' object and the actions (behavior) that are associated with it. Objects can contain other objects. A key part of the definition of an object is unique identity. In an object-oriented system, each object has a unique system-wide identifier (the OID) that is independent of the values of its attributes and, ideally, invisible to the user.
- Methods define the behavior of the object. They can be used to change the object's state by modifying its attribute values or to query the value of selected attributes. Messages are the means by which objects communicate. A message is simply a request from one object (the

sender) to another object (the receiver) asking the second object to execute one of its methods. The sender and receiver may be the same object.

- Objects that have the same attributes and respond to the same messages can be grouped together to form a class. The attributes and associated methods can then be defined once for the class rather than separately for each object. A class is also an object and has its own attributes and methods, referred to as class attributes and class methods, respectively. Class attributes describe the general characteristics of the class, such as totals or averages.
- Inheritance allows one class to be defined as a special case of a more general class. These special cases are known as subclasses and the more general cases are known as superclasses. The process of forming a superclass is referred to as generalization; forming a subclass is specialization. A subclass inherits all the properties of its superclass and additionally defines its own unique properties (attributes and methods).

All instances of the subclass are also instances of the superclass. The principle of substitutability states that an instance of the subclass can be used whenever a method or a construct expects an instance of the superclass.

- Overloading allows the name of a method to be reused within a class definition or across definitions. Overriding, a special case of overloading, allows the name of a property to be redefined in a subclass. Dynamic binding allows the determination of an object's type and methods to be deferred until runtime.
- In response to the increasing complexity of database applications, two 'new' data models have emerged: the Object-Oriented Data Model (OODM) and the Object-Relational Data Model (ORDM). However, unlike previous models, the actual composition of these models is not clear. This evolution represents the third generation of DBMSs.

Review questions

13.1 Discuss the general characteristics of advanced database applications.

13.2 Discuss why the weaknesses of the relational data model and relational DBMSs may make them unsuitable for advanced database applications.

13.3 Define each of the following concepts in the context of an object-oriented data model:

- (a) Abstraction, encapsulation, and information hiding;
- (b) Objects and attributes;

- (c) Object identity;
- (d) Methods and messages;
- (e) Classes, subclasses, superclasses, and inheritance;
- (f) Overriding and overloading;
- (g) Polymorphism and dynamic binding.

13.4 Discuss the difficulties involved in mapping objects created in an object-oriented programming language to a relational database.

13.5 Describe the three generations of DBMSs.

13.6 Describe how relationships can be modeled in an OODBMS.

13.7 Describe the different modeling notations in the UML.

Chapter 14: Introduction to data warehousing

General Objectives

At the end of this chapter you will be able to: Explain How data warehousing evolved and the main concepts and benefits associated with data warehousing.

Specific objectives:

At the end of this chapter you will be able to:

- Explain how Online Transaction Processing (OLTP) systems differ from data warehousing.
- Identify the problems associated with data warehousing.
- Discuss the architecture and main components of a data warehouse.
- Explain the important data flows or processes of a data warehouse.
- Discuss the main tools and technologies associated with data warehousing.
- Explain the issues associated with the integration of a data warehouse and the importance of managing metadata.
- Explain the concept of a data mart and the main reasons for implementing a data mart.
- Explain the main issues associated with the development and management of data marts.

Chapter Summary

St. Mary's University
Faculty of Informatics

- Data warehousing is subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process. A data warehouse is data management and data analysis technology.
- Data Web house is a distributed data warehouse that is implemented over the Web with no central data repository. The potential benefits of data warehousing are high returns on investment, substantial competitive advantage, and increased productivity of corporate decision-makers.
- A DBMS built for Online Transaction Processing (OLTP) is generally regarded as unsuitable for data warehousing because each system is designed with a differing set of requirements in mind. For example, OLTP systems are design to maximize the transaction processing capacity, while data warehouses are designed to support *ad hoc* query processing.
- The major components of a data warehouse include the operational data sources, operational data store, load manager, warehouse manager, query manager, detailed, lightly and highly summarized data, archive/backup data, metadata, and end-user access tools.
- The operational data source for the data warehouse is supplied from mainframe operational data held in first generation hierarchical and network databases, departmental data held in proprietary file systems, private data held on workstations and private servers and external systems such as the Internet, commercially available databases, or databases associated with an organization's suppliers or customers.
- The operational data store (ODS) is a repository of current and integrated operational data used for analysis. It is often structured and supplied with data in the same way as the data warehouse, but may in fact simply act as a staging area for data to be moved into the warehouse. The load manager (also called the *frontend* component) performs all the operations associated with the extraction and loading of data into the warehouse. These operations include simple transformations of the data to prepare the data for entry into the warehouse.
- The warehouse manager performs all the operations associated with the management of the data in the warehouse. The operations performed by this component include analysis of data to ensure consistency, transformation and merging of source data, creation of indexes and views, generation of denormalizations and aggregations, and archiving and backing-up data.

- The query manager (also called the *backend* component) performs all the operations associated with the management of user queries. The operations performed by this component include directing queries to the appropriate tables and scheduling the execution of queries.
- End-user access tools can be categorized into five main groups: data reporting and query tools, application development tools, executive information system (EIS) tools, Online Analytical Processing (OLAP) tools, and data mining tools.
- Data warehousing focuses on the management of five primary data flows, namely the inflow, up-flow, down-flow, outflow, and meta-flow.
- Inflow is the processes associated with the extraction, cleansing, and loading of the data from the source systems into the data warehouse.
- Up-flow is the processes associated with adding value to the data in the warehouse through summarizing, packaging, and distribution of the data.
- Down-flow is the processes associated with archiving and backing-up of data in the warehouse.
- Outflow is the processes associated with making the data available to the end-users.
- Meta flow is the processes associated with the management of the metadata (data about data).
- Data mart is a subset of a data warehouse that supports the requirements of a particular department or business function. The issues associated with data marts include functionality, size, load performance, users 'access to data in multiple data marts, Internet/intranet access, administration, and installation.

Review Questions

- 14.1 Define the terms data warehousing and data mining.
- 14.2 What is data warehousing and why do we need it?
- 14.3 What are the rules for data warehouses?

Chapter 15 Introduction to Data Mining

General objectives

At the end of this chapter you will be able:

Explain the concepts associated with data mining.

Specific objectives

- Explain the data mining process and its operational techniques
- Identify the main features of data mining operations, including predictive modeling, database segmentation, link analysis, and deviation detection.
- Discuss the techniques associated with the data mining operations.
- Identify important characteristics of data mining tools.

Chapter Summary

- **Data mining** is the process of extracting valid, previously unknown, comprehensible, and actionable information from large databases and using it to make crucial business decisions.
- There are four main operations associated with data mining techniques: predictive modeling, database segmentation, link analysis, and deviation detection.
- Techniques are specific implementations of the operations (algorithms) that are used to carry out the data mining operations. Each operation has its own strengths and weaknesses.
- **Predictive modeling** can be used to analyze an existing database to determine some essential characteristics (model) about the data set. The model is developed using a supervised learning approach, which has two phases: training and testing. Applications of predictive modeling include customer retention management, credit approval, cross-selling, and direct marketing. There are two associated techniques: classification and value prediction.
- **Database segmentation** partitions a database into an unknown number of segments, or clusters, of similar records. This approach uses unsupervised learning to discover homogeneous sub-populations in a database to improve the accuracy of the profiles.
- **Link analysis** aims to establish links, called associations, between the individual records, or sets of records, in a database. There are three specializations of link analysis: associations discovery, sequential pattern discovery, and similar time sequence discovery. Associations discovery finds items that imply the presence of other items in the same event. Sequential pattern discovery finds patterns between events such that the presence of one set of items is followed by another set of items in a database of events over a period of time. Similar time sequence discovery is used, for example, in the discovery of links between two sets of data that are time-dependent, and is based on the degree of similarity between the patterns that both time series demonstrate.

- **Deviation detection** is often a source of true discovery because it identifies outliers, which express deviation from some previously known expectation and norm. This operation can be performed using *statistics* and *visualization* techniques or as a by-product of data mining.
- The Cross Industry Standard Process for Data Mining (**CRISP-DM**) specification describes a data mining process model that is not specific to any particular industry or tool.
- The important characteristics of data mining tools include: data preparation facilities; selection of data mining operations (algorithms); scalability and performance; and facilities for understanding results.
- A data warehouse is well equipped for providing data for mining as a warehouse not only holds data of high quality and consistency, and from multiple sources, but is also capable of providing subsets (views) of the data for analysis and lower level details of the source data, when required.

Review Questions

15.1 Discuss what data mining represents.

15.2 What Can Data Mining Do?

15.3 Describe how the following data mining operations are applied and provide typical examples for each:

(a) Predictive modeling, (b) Database segmentation, (c) Link analysis, (d) Deviation detection

15.4 Describe the main aims and phases of the CRISP-DM model.

15.5 Discuss the relationship between data warehousing and data mining.

Answers to selected questions

Chapter 1

1.1 (b) Database is a shared collection of logically related data designed to meet the information needs of an organization. Since it is a shared corporate resource, the database is integrated with minimum amount of or no duplication.

1.2 (c) A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise.

Chapter 2

2.2 The three levels of ANSI-SPARC model are

- i. Internal level The physical representation of the database on the computer. This level describes *how* the data is stored in the database.
- ii. Logical data independence Logical data independence refers to the immunity of the external schemas to changes in the conceptual schema.
- iii. Physical data independence Physical data independence refers to the immunity of the conceptual schema to changes in the internal schema.

2.3 Data model is an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.

2.7 The major components of DBMS are

- Query processor ,Database manager (DM) , File manager ,DML preprocessor ,DDL compiler
- Catalog manager

Chapter 3

3.1 (a) A relation is a table with columns and rows.

(b) An attribute is a named column of a relation.

(c) A domain is the set of allowable values for one or more attributes

(e)The degree of a relation is the number of attributes it contains. The cardinality of a relation is the number of tuples it contains.

3.3 A relation has the following properties:

- the relation has a name that is distinct from all other relation names in the relational schema;
- each cell of the relation contains exactly one atomic (single) value;
- each attribute has a distinct name;
- the values of an attribute are all from the same domain;
- each tuple is distinct; there are no duplicate tuples;
- the order of attributes has no significance;
- the order of tuples has no significance, theoretically. (However, in practice, the order may affect the efficiency of accessing tuples.)

Chapter 5

5.1 The two major components are:

- Data Definition Language (DDL) for defining the database structure and controlling access to the data;
- Data Manipulation Language (DML) for retrieving and updating data.

Chapter 6

6.1 Database security is The mechanisms that protect the database against intentional or accidental threats.

6.2 The main the types of threat (i) Theft and fraud (ii). Loss of confidentiality (iii). Loss of privacy (iv). Loss of integrity (v). Loss of availability

Chapter 7

7.1 Transaction is An action, or series of actions, carried out by a single user or application program, which reads or updates the contents of the database.

7.9 Timestamp is a unique identifier created by the DBMS that indicates the relative starting time of a transaction.

Chapter 8

8.1 Inheritance allows one class to be defined as a special case of a more general class. These special cases are known as **subclasses** and the more general cases are known as **superclasses**.

8.2 The process of forming a superclass is referred to as **generalization** and the process of forming a subclass is **specialization**

Chapter 9

9.1 Superkey: an attribute or set of attributes that uniquely identifies a tuple within a relation.

Candidate key: A superkey such that no proper subset is a superkey within the relation

Primary key : The candidate key that is selected to identify tuples uniquely within the relation

Foreign key :an attribute, or set of attributes, within one relation that matches the candidate key of some (possibly the same) relation.

9.3 An entity type is referred to as being **strong** if its existence does not depend upon the existence of another entity type.

Weak entity type An entity type that is existence-dependent on some other entity type.

Chapter 11

11.1 **Query processing** the activities involved in parsing, validating, optimizing, and executing a query.

11.4 The typical stages of **query decomposition** are analysis, normalization, semantic analysis, simplification, and query restructuring.

Chapter 12

12.1 Distributed database A logically interrelated collection of shared data (and a description of this data) physically distributed over a computer network.

Distributed DBMS (DDBMS) :The software system that permits the management of the distributed database and makes the distribution transparent to users.

12.6 Communication networks may be classified in several ways. One classification is according to whether the distance separating the computers is short (local area network) or long (wide area network). A **local area network** (LAN) is intended for connecting computers over a relatively short distance, for example, within an office building, a school or college, or home. Sometimes one building will contain several small LANs and sometimes one LAN will span several nearby buildings. LANs are typically owned, controlled, and managed by a single organization or individual.

A **wide area network** (WAN) is used when computers or LANs need to be connected over long distances. The largest WAN in existence is the Internet.

Chapter 13

13.2 The weaknesses of relational DBMSs are

- i. Poor representation of 'real world' entities
- ii. Semantic overloading
- iii. Poor support for integrity and enterprise constraints
- iv. Homogeneous data structure
- v. Limited operations
- vi. Difficulty handling recursive queries
- vii. Impedance mismatch

- viii. Other problems with RDBMSs associated with concurrency,
- ix. schema changes, and poor navigational access

Chapter 14

14.1 Data warehousing A subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision-making process.

Data mining The process of extracting valid, previously unknown, comprehensible, and actionable information from large databases and using it to make crucial business decisions.

Data mining discovers information within data warehouses that queries and reports cannot effectively reveal.

Chapter 15

15.1 Data mining, or knowledge discovery, is the computer-assisted process of digging through and analyzing

enormous sets of data and then extracting the meaning of the data

15.2 Although data mining is still in its infancy, companies in a wide range of industries – including finance,

health care, manufacturing, transportation, are already using data mining tools and techniques to take advantage of historical data

Suggested Readings

1. Database systems Thomas M. Connolly • Carolyn E. Begg, University of Paisley A Practical Approach to Design, Implementation, and Management (4th Edition)
2. Database Management Systems, Ramakrishnan. 2003 (3rd edition)
3. Database Processing, David M. Western Washington University, 2012 (12th edition)
4. Modern Database management Jeffery A. 2007 (8th edition)
5. Database Systems for Management, James F. 2010 (3rd edition)
6. Fundamental of Relational Database Management System Sumathi S. volume 47, 2007
7. The Relational Model for Database Management, E.F. Codd, Version 2, 1999
8. Fundamentals of Database System, Ramez Elmasri, Shamkant B. Navathe. 1994. (2nd edition)
9. Databases: Design, development, and deployment McGraw-Hill, N.Delhi, 2001

St. Mary's University
Faculty of Informatics

10. An Introduction to Database System, Bipin C. Desai, Galgotia Publication 2002
11. Distributed Database Management Systems, Saeed K. IEEE Computer Society 2010.
12. Grid and Cloud Database Management, Sandro F. 2011
13. Multimedia Database Management Systems, Guojun Lu. 1999

Study Guide

For

Formal Language Theory and Compiler Design and Analysis

Department of Computer Science
Faculty of Informatics



ቅድስት ማርያም ዩኒቨርሲቲ
St. Mary's University, Ethiopia

Prepared by
Sebsibe Hailemariam (PhD)

January 2012

Summary

Computer Science deals with the theoretical and practical foundations of computation as well as their implementation in computer system so as to address critical and vital issues of the discipline. Computer Science gives more emphasis to language representation and analysis among the various issues that the field is concerned on. Language as a means of communication and formalism enables standard mechanism of data representation and processing that plays major role in the successful trends in computing discipline.

Formal language is a mathematical formulation of language representation which is the most appropriate model to be adopted for computational purpose. Compiler is a language translator which translates a set of statement organized in some high level programming language into machine language that can be executed by the computing devices. These courses are corner stone to Computer Science students as they provide diverse insight into the field.

This guideline is not a reference that students should use during the due time of the courses formal language and compiler design. Rather it can be used for the students as a checklist to see what is expected from the students to know and what they already know. Hence, this guideline clearly shows the students the existence of any gap among the expected knowledge, skill and attitude to be achieved at the end of the course and the reality.

This guideline is also relevant not only to students but also for instructors that teaches these courses. Instructors should use this guideline as a check list to periodically assess their status while teaching the course and plan a head how to cover the important topics identified and needed to be known by the students.

In addition, the guideline is an important tool for anyone who seeks to study these courses independently. Finally, this guideline is an important asset to the Department to give a direction to instructors what needs to be covered while he/she is teaching the course.

PART I

Formal Language Theory

Summary

Formal language is theoretical definition of languages and tools that define them. Formal language is the basis to study the behavior of languages, machines that define some languages. Tools such as regular expressions, finite state automata, finite state transducer, context free grammars, pushdown automata, Turing machine and others are studied well using these concepts. Natural languages processing is attempted as formal language theory nowadays get more advancement as natural languages are fairly modeled using the formal languages as a modeling tool.

This part of the guideline deals more about the introductory concepts of formal language; finite state automata (deterministic and non-deterministic); regular expressions; context free grammar; pushdown automata language; and languages defined by these concepts.

Unit one

Fundamentals of Formal languages

Summary

Formal language is a set of strings where the strings are formed from sequences of alphabets. Most operations defined in formal language theory are operations of a set. Set theory is the building block of formal language theory. Hence, this unit is intended to provide a guideline for the students on this introductory concept of formal language theory.

General Objective

The general objective of this chapter is to revise the basic concepts for the foundation of formal language theory such as set theory, graph theory, formal proof methods and searching and traversing graphs.

Specific Objectives

At the end of this unit, students should be able to:

- Define a set
- Identify different notations to represent set
- Distinguish complete listing method, partial listing method, and set builder method.
- Describe the power set of a set
- Distinguish finite versus infinite set
- Describe the union, intersection, difference and complement operations
- Describe the subset, equivalence, proper subset of a set
- Describe the commutative, associative, distributive properties of a set
- Describe the De Morgan's law
- Describe the formal proof methods
- Describe proof by deduction
- Use proof by deduction
- Describe proof by induction
- Use proof by induction
- Describe proof by contradiction
- Use proof by contradiction
- Describe the terms alphabets, string, language in formal language analysis

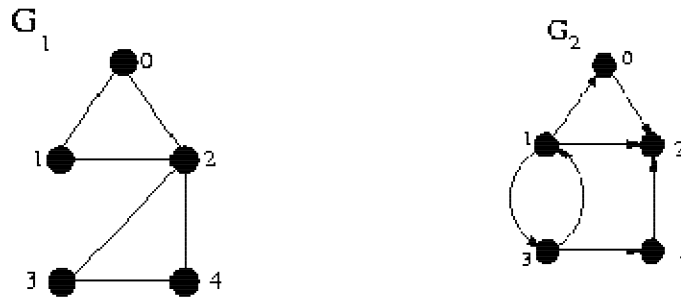
- Identify empty string
- Describe union operation in a string
- Describe concatenation operation in a string
- Describe exponentiation operation in a string
- Describe Kleene closure operator of a set of strings/symbols
- Describe basics of graphs
- Distinguish order of a graph and size of a graph
- Distinguish between cycle graph, cycle in a graph, loop in a graph
- Distinguish between complete graph and connected graph
- Distinguish di-graph from a graph
- Identify degree of a graph
- Identify in-degree, out-degree, degree of a di-graph
- Distinguish a walk, path, cycle in a graph
- Describe disconnected components
- Describe graph representation techniques (Adjacency matrix and Adjacency list)
- Describe graph traversal methods (depth first and breadth first approach)

Self-test Questions without answer key

Attempt all of these questions and make sure that you answer them correctly.

1. Give an example of a set that can be represented using complete listing method
2. Give an example of a set that can be represented using partial listing method
3. Give an example of a set that can be represented using set builder method
4. An infinite set can be represented using complete listing method (TRUE or FALSE)
5. $A \subseteq B$, if and only if all elements of A are also elements of B (TRUE or FALSE)
6. $A \subseteq B$, if and only if all elements of A are also elements of B (TRUE or FALSE)
7. Two sets A and B are said to be equivalent if and only if all elements of A are in B (TRUE or FALSE)
8. Given two set A and B each having N and M elements each, what are the maximum and minimum number of elements in $A \cup B$.

9. Given two set A and B each having N and M elements each, what are the maximum and minimum number of elements in $A \cap B$.
10. What will be $A \cup B$ and $A \cap B$ given $A = \{1, 2, 3, 8, 11\}$ and $B = \{3, 2, 1, 9, 8\}$
11. What will be $A - B$ given $A = \{1, 2, 3, 8, 11\}$ and $B = \{3, 2, 1, 9, 8\}$
12. What will be the complements of A given $A = \{1, 2, 3, 8, 11\}$ and $U = \{1, 2, 3, \dots, 15\}$
13. Proof by deduction that the roots of the equation $x^2 - 5x + 6 = 0$ are +2 and +3
14. Proof by deduction that the number of vertices with odd number of degree in a graph are even
15. Proof by induction that the sum of the integers from 1 to N equals $(N+1)*N/2$
16. The set of ASCII characters of a computer defines an alphabet in formal language theory (TRUE or FALSE)
17. Any string in a natural language are defined from the Latin alphabets $\{a, b, c, \dots, z, A, B, C, \dots, Z\}$
18. Define at least three different formal languages from the alphabet $\{a, b, c, d, e\}$
19. Differentiate $\{\}$, $\{\varepsilon\}$ and ε from formal language perspective
20. Given the two graphs G1 and G2 shown below, answer the following questions from A-Z



- a. Identify the degree of all the vertices of G1 and G2
- b. Identify the in-degree and out-degree of G2
- c. Identify the degree of the graph G1 and G2
- d. Distinguish order of a graph and size of a graph
- e. Find cycles from G1 and G2
- f. Find a walk from G1 which is not a path
- g. Is there a loop in G1 and G2

- h. How many connected component G1 has
- i. Represent G1 and G2 using Adjacency matrix
- j. Represent G1 and G2 using Adjacency list

Questions with answer key

1. Given a set $A = \{C++, C, C\# \}$, list all the power set of A ($P(A)$). How many elements does $P(A)$ has?
2. According to the De Morgan's law, given A as the set of all even integers, B as the set of integers which are multiple of three and U as the set of all integers answer the following questions
 - a. Describe what the complement of $A \cup B$ be verbally as well as using partial listing method
 - b. Describe what the complement of $A \cap B$ be verbally as well as using partial listing method
3. Proof by induction that the sum of the degrees of a graph is twice the number of edges

Answer key

1. The number of elements $P(A)$ will have is exactly 2^n where n is the number of elements of A. Hence A has 8 elements in its power set. These are
 $\{\emptyset, \{C++\}, \{C\}, \{C\# \}, \{C++, C\}, \{C++, C\# \}, \{C, C\# \}, \{C++, C, C\# \}$
2.
 - a. $A \cup B$ is the set of all even integers and all odd integers which are multiple of 3. Hence, the complement of $A \cup B$ becomes a set of non-even integer which are not also multiple of three. Therefore, $(A \cup B)' \equiv$ The set of odd integers which are not multiple of three
 - b. $A \cap B$ is the set of integers which are both even and multiple of three. That means, the complement of $A \cap B$ becomes a set of all non-even integer or integers which are not multiple of three. Hence, $(A \cap B)' \equiv$ The set of odd integers or integers which are not multiple of three

3. Proof by induction that the sum of the degrees of a graph is twice the number of edges

Base phase: assume there is only one edge in the graph. A single edge connects two vertices only say vertices A and B. Hence, the degree of all vertices becomes 0 other than A and B which have a degree of 1 each. Hence the sum of all the degrees becomes $1 + 1 + 0 + 0 + \dots + 0 = 2$. Hence it is true for $N=1$.

Induction phase: Assume it is true for any N such that $N \leq m$, we want to show that it is also true for $N= m+1$ (i.e the sum of all the degrees of the vertices becomes $2(m+1) = 2m + 2$)?

Let's assume we have a graph G of $m+1$ edge where one of the edges is from A to B. If we ignore this edge, we will have a Graph G' with m edge. As G' has m edges, then the sum of the degrees of all the vertices become $2m$ (induction assumption above). If we add back the removed edge that links A with B, G' become modified into G and degree of A and B will increase by one each. Hence, the total sum of degrees of all vertices of G becomes the total sum of degrees of G' plus 2 (i.e. $2m + 2 = 2(m+1)$).

Unit Two

Finite State Automata and Regular Languages

Summary

One way to define a language is to construct an automaton: a kind of abstract computer that takes a string as input and produces a yes-or-no answer. The language it defines is the set of all strings for which it says yes.

The simplest kind of automaton is the finite state automaton. More complicated automata have some kind of unbounded memory to work with or use another structure such as stack; in effect, they will be able to grow to whatever size necessary to handle the input string they are given or the stack will be used to memorize important past event happened. PDA and TM have got stack and tape as memory which are infinite but the states in both cases are finite. But this chapter focuses on introducing students about finite state automata (both deterministic and nondeterministic) and the concept of regular languages. A finite automaton has a finite memory that is fixed in advance (finite states) and doesn't use stack memory. Whether the input string is long or short, complex or simple, the finite automaton must reach its decision using the same fixed and finite memory.

General Objective

The purpose of this unit is to introduce the concepts of deterministic finite state automata, regular languages, non-deterministic Finite State Automata (FSA) and their relationships.

Specific Objectives

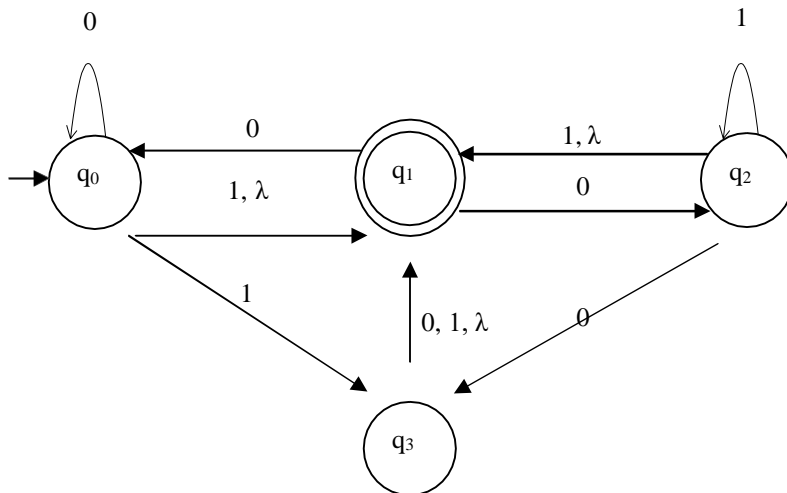
At the end of this unit, students should be able to:

- Identify the concept of states and its interpretation
- Describe what a label refers to in a transition from one state to another state
- Describe FSA in terms of set of states and transitions
- Distinguish start state, final state, and other states
- Describe what do we mean by a string is accepted by a FSA
- Explain a language accepted by the FSA automata
- Identify Deterministic Finite state Automata (DFSA)
- Illustrate DFSA as a five-tuple machine

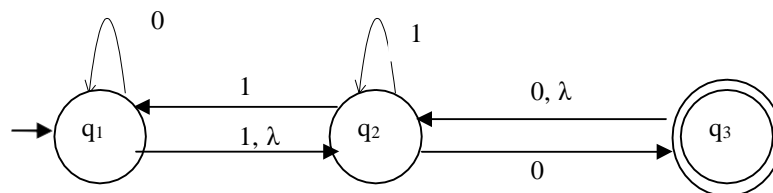
- Describe what transition function be for DFSA
- Describe extended transition function of DFSA
- Provide the definition for the extended transition function of DFSA
- Explain what do we mean by a DFSA machine M accepts a string x
- Define a language defined by a given DFSA machine M
- Define Regular Language
- Identify the closure properties of regular languages
- Describe and prove the closure properties
- Prove that intersection of two regular language is also regular language
- Prove that union of two regular language is also regular language
- Prove that concatenation of two regular language is also regular language
- Prove that Kleene closure of a regular language is also regular language
- Identify Non-deterministic Finite state Automata (NFSA)
- Distinguish NFSA from DFSA
- Illustrate spontaneous transition (ϵ -**Transition**) in NFSA
- Illustrate NFSA as a five-tuple machine
- Describe how to construct union/concateration of two NFSA to form complex NFSA
- Describe what transition function be for NFSA
- Describe an instantaneous description of an NFSA
- Describe extended transition function of NFSA
- Provide the definition for the extended transition function of NFSA
- Explain what do we mean by an NFSA machine M accepts a string x
- Define a language defined by a given NFSA machine M
- Explain how to convert DFSA machine into NFSA
- Explain how to convert NFSA machine into DFSA
- Identify the pros and cons of DFSA over NFSA
- Identify the pros and cons of NFSA over DFSA

Self-test Questions

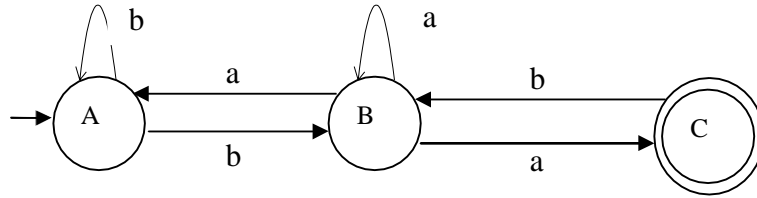
1. Define the language defined by the DFSA shown above
2. Find the DFSA that accept a string **w** of the alphabet **{a,b}** where **$N_a \bmod 3 > 1$** . **N_a** is the number of the symbol **a** in **w**.
3. Answer the following questions based on the following FSA.



- A. What are Q , Σ and F ?
 - B. Is the string 0000111 accepted by the FSA? Justify your answer.
 - C. Convert the FSA to an NFSA without **ϵ -transition** and then check the acceptance of the string 0101.
 - D. Convert the NFSA without **ϵ -transition** moves you obtained above to the corresponding DFSA and then check the acceptance of the string 1100.
 - E. Define the language determined by the machine
4. Construct a DFA or NFA that generates a string over the alphabet **{a,b}** which has **m** a's followed by **n** b's for all **n** and **m** greater than or equals to 0
 5. Discuss the similarities and differences between DFSA and NFSA.
 6. Find an equivalent NFSA without λ corresponding to the NFSA shown in the figure below.



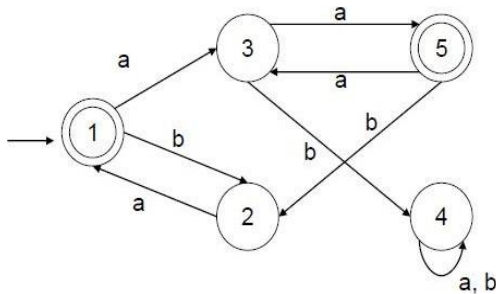
7. Find an equivalent DFSA for the following NFSA.



8. Define the language determined by the NFSA machine above

9. Find a regular grammar that generates a language that is accepted by the automaton given in question 11

10. Construct a minimal DFSA for the following DFSA.



11. Given an FSA that accepts binary representation of an integer divisible by three.

12. Given an FSA that accepts binary representation of an integer that ends with 01.

13. Define extended transitions for a DFA machine

14. Describe acceptance a string w by DFS using extended transitions concept

15. Define extended transitions for a NFA machine

16. Describe acceptance a string w by NFS using extended transitions concept

17. Prove that any DFA machine can be converted into NFA machine

18. Prove that any NFA machine can be converted into DFA machine

Questions with answer key

1. Find a DFSA that accept a string w of the alphabet $\{a\}$ where $|w| \bmod 3 \neq |w| \bmod 2$.
2. Show that the language $L = \{a^n: n \geq 3\}$ is regular.

3. Design a DFA that accept only a number (in base 10) which is multiple of 3.

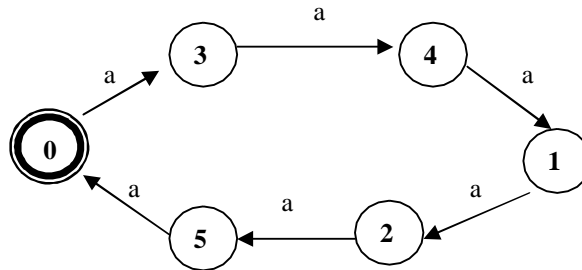
Answer key

1. To answer this question, we may start by defining a state as an indicator that indicates the remainder for the modulo for the current length when divided by two and three. Hence, I define states as follows:

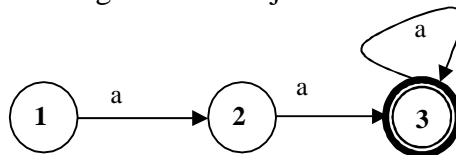
- a. State 0 as a state when length modulo 2 = 0 and length modulo 3 = 0*
- b. State 1 as a state when length modulo 2 = 1 and length modulo 3 = 0*
- c. State 2 as a state when length modulo 2 = 0 and length modulo 3 = 1*
- d. State 3 as a state when length modulo 2 = 1 and length modulo 3 = 1*
- e. State 4 as a state when length modulo 2 = 0 and length modulo 3 = 2*
- f. State 5 as a state when length modulo 2 = 1 and length modulo 3 = 2*

Hence, except state 0 and 3 are accepting states as per the description.

The following DFSA demonstrate the above argument



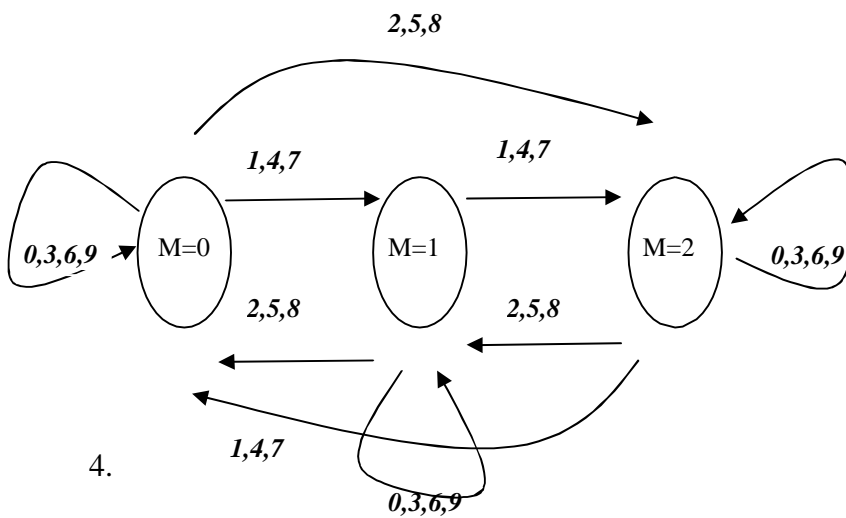
2. In order to show the language $L = \{a^n : n \geq 3\}$ is regular, we need to construct a DFA that will accept all strings of L and reject others. The following DFA fulfill the requirement



3. $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$. A digit 0, 3, 6 and 9 are all multiple of 3 hence their modulo to 3 is 0. When we compute modulo of the digit 1, 4, 7, 10 to 3, it will be 1 and the remaining digits (2, 5, and 8) has module of 2. If $X \bmod 3$ is M for any integer X then $Y \bmod 3$ becomes N where Y is X concatenated with some digit d . This relations is shown in the table below for a given X

$M = X \bmod 3$	Possible d value	$N = Y \bmod 3$
0	0,3,6,9	0
0	1,4,7	1
0	2,5,8	2
1	0,3,6,9	1
1	1,4,7	2
1	2,5,8	0
2	0,3,6,9	2
2	1,4,7	0
2	2,5,8	1

Hence the DFA becomes:



Unit Three

Regular Expressions and Regular Languages

Summary

In the previous unit, study guide were prepared on finite state automata machine that demonstrate both deterministic and non-deterministic finite state automata. In this unit, one of the most powerful mechanisms to define regular language, which is a regular expression, will be presented. Regular expressions are getting popularity in various programming languages and operating system shell commands to process natural languages. Regular expressions will have the same expression power of DFSA and NFSA and hence can be used to express a language called regular language.

General Objective

The general objective of this unit is to guide students on topics such as definition and concepts of regular expression formation, language defined by regular expression, conversion of NFSA and DFSA into regular expression and vice versa and its applications.

Specific Objectives

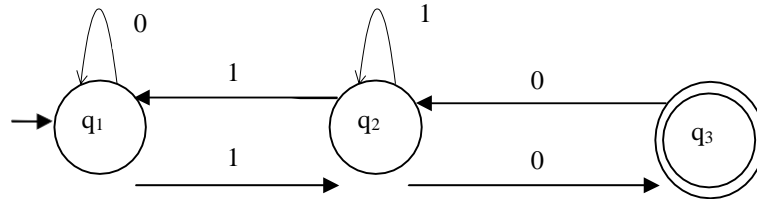
At the end of this unit, students should be able to:

- Define a regular expression
- Define primitive regular expressions
- Describe the operators used in regular expressions which include union (+), concatenation (.), and Kleene closure (*)
- Define complex regular expressions from the primitive ones using the operators
- Define the language that a regular expression denotes
- Show that the language that a regular expression denotes is a regular language by constructing NFSA that accept the languages denoted by primitive regular expressions and complex regular expressions
- Find a regular expression that denotes the language accepted by NFSA
- Know the existence of families of languages that are not regular
- Define Pumping Lemma for regular languages
- Show that some languages are not regular languages by using Pumping Lemma

- Distinguish between the formal regular expression definition and the various regular expression defined by Perl, Python, Java, Unix shell command grep/egrep, etc

Self-test Questions

1. Find an NFSA that accepts $L(r)$ where $r = aa^*(a + b)$
2. Find a regular expression corresponding to the NFSA shown in the figure below.

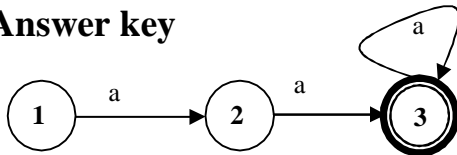


3. State pumping lemma for regular languages
4. Using pumping lemma show that the language $L = \{a^n b 2^n \mid n > 0\}$ is not regular.
5. Write regular expression for the languages all strings ending in 01 given $\Sigma = \{0, 1\}$.
6. Write regular expression for the languages all strings not ending in 01 given $\Sigma = \{0, 1\}$.
7. Find regular expressions for the languages on $L = \{w: n_a(w) \bmod 3 = 0\}$ given $\Sigma = \{a, b\}$.
Find regular expressions for the languages on $L = \{w: n_a(w) \bmod 5 > 0\}$ given $\Sigma = \{a, b\}$.
8. State the precedence rules for the regular expression operators
9. Construct a regular expression r for C++ identifiers (i.e $x \in L(r) \Leftrightarrow x$ is a C++ identifier)
10. Convert the regular expressions $(0+1)01$ into NFA with ϵ transitions
11. Convert the regular expressions $00(0+1)^*$ into NFA with ϵ transitions
12. Prove the following theorems
 - a. The union of two regular language is regular
 - b. The intersection of two regular language is regular
 - c. The complement of a regular language is regular
 - d. The difference of two regular language is regular
 - e. The reversal of a regular language is regular
 - f. The closure (star) of a regular language is regular
 - g. The concatenation of two regular language is regular

Questions with answer key

1. Convert the regular expressions 01^* into NFA with ε transitions
2. Find regular expressions for the languages on $L = \{w: |w| \bmod 3 = 0\}$ given $\Sigma = \{a, b\}$.
3. Write regular expression for the languages all strings containing even number of zeros given $\Sigma = \{0, 1\}$.

Answer key



Unit Four

Context Free Grammar

Summary

So far we have seen finite state automata and regular expression as a mechanism to define regular language and their properties. In this chapter we will look into more complex means of language construction tool called context free grammar. It is a means to define grammar of a language. Context free grammar defines set of strings using set of production rules. It is called context free because each production are independent from each other.

General Objective

The general objective of the chapter is to deal with CFG and the language defined by the grammar. Moreover, links will be established with the concepts that we have discussed so far.

Specific Objectives

At the end of this unit, students should be able to:

- Define what a Context Free Grammar (CFG) is
- Describe role of Context Free Grammar for parsing
- Identify whether a given grammar is a CFG or not
- Define what a Context Free Language (CFL) is
- Define what a derivation or a parse tree is
- Explain how parse tree is generated from a CFG and token sequence
- Identify derivation in a CFG
- Define leftmost derivation
- Define rightmost derivation
- Define extended derivation (n-step derivation)
- Identify single step grammar
- Identify left linear grammar
- Identify right linear grammar
- Identify ambiguous grammar
- Describe a language defined by a grammar

- Define a string from $L(G)$ using extended derivation
- Identify unambiguous grammar
- Show how CFGs can be used in parsing arithmetic expression
- Explain how operator derived ambiguity can be resolved
- Identify what left recursion (immediate and non-immediate left recursion)
- Explain how to remove left recursion
- Explain the purpose of left factoring
- Explain how to do left factoring
- Define λ -productions
- Define nullable nonterminals
- Explain how to remove λ -productions
- Define unit productions
- Explain how to remove unit productions
- Define useless/irrelevant productions
- Explain how to remove irrelevant productions
- Define normal forms
- Define Chomsky Normal Form (CNF)
- Show how a λ -free CFG can be converted into an equivalent CFG in CNF
- Define Greibach Normal Form (GNF)
- Show how a λ -free CFG can be converted into an equivalent CFG in GNF
- Define Pumping Lemma for CFLs
- Show that some languages are not CFLs by using Pumping Lemma for CFLs
- Describe membership algorithm for CFLs
- Show CFLs are closed under union, concatenation and Kleene closure, not intersection
- Show that every regular languages have a CFG
- Show that all CFG do not define regular language
- Show that every single step grammar define regular language
- Show that every right linear grammar define regular language
- Show that every left linear grammar define regular language

Self-test Questions

- Describe the following grammars using the different grammar characteristics (context free or not, the language it defines, linearity, ambiguity, single step or not, etc)

B. $S \rightarrow aSb \mid \lambda$

B: $S \rightarrow aB \mid A$
 $aA \rightarrow aA \mid a \mid CBA$

C: $S \rightarrow aB$

$B \rightarrow \lambda$

$B \rightarrow bA \mid b$

D: $A \rightarrow aB$

$A \rightarrow a$

$B \rightarrow cA \mid C$

$C \rightarrow cC \mid c$

- Given a grammar $G = (N, T, P, S)$ with productions:

$S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

And a string $x = aaabbb$

A) find a left most and right most derivations for x .

B) draw the parse tree for x .

C) is the grammar ambiguous?

- Given a grammar $G = (N, T, P, S)$ with productions:

$S \rightarrow SbS \mid ScS \mid a$

and a string $x = abaca$

A) find a left most and right most derivations for x .

B) draw the parse tree for x .

C) is the grammar ambiguous?

- Consider the following grammar:

$E \rightarrow T \mid E + T \mid E - T$

$T \rightarrow F \mid T * F \mid T / F$

$F \rightarrow a \mid b \mid c \mid (E)$

Draw parse trees for the following strings

a) $a*b+c$

b) $a+b*c$

c) $(a+b)*c$

d) $a-b-c$

5. Given CFG grammar G with productions

$S \rightarrow aS \mid AB,$

$A \rightarrow \lambda,$

$B \rightarrow \lambda,$

$D \rightarrow b,$

construct a grammar G' without null productions such that $L(G') = L(G) - \{\lambda\}$.

6. Remove all unit productions from the following CFG.

$S \rightarrow Aa \mid B$

$B \rightarrow A \mid bb$

$A \rightarrow a \mid bc \mid B$

7. Eliminate useless productions from

$S \rightarrow a \mid aA \mid B \mid C$

$A \rightarrow aB \mid \lambda$

$B \rightarrow Aa$

$C \rightarrow cCD$

$D \rightarrow ddd$

8. Convert the grammar with productions

$S \rightarrow ABa \mid ABS$

$A \rightarrow aab \mid abc$

$B \rightarrow Ac$

to Chomsky normal form.

9. Convert the grammar with productions

$S \rightarrow abSb \mid aa$

to Greibach normal form.

10. Define a context free grammar for declaration statement for a C++ programming language

Unit Five

Pushdown Automata

Summary

Push down automata is a type of machine that defines context free grammar language. Unlike finite state automata, pushdown automata is an extension to the NDFSA with addition stack that one can push, pop and read from the stack.

General Objective

The general objective the chapter is to introduce pushdown automata and its functionality.

Specific Objectives

At the end of this unit, students should be able to:

- Define Pushdown Automata (PDA)
- Illustrate PDA as a 7-tuple automata
- Describe what happens when a PDA makes a move
- Show the association between PDA and CFL
- Show the differences between FSA and PDA
- Show how PDA accepts language families that cannot be accepted by FSA
- Describe the types of PDA: deterministic PDA (DPDA) and nondeterministic PDA (NPDA)
- Describe an NPDA using transition diagram
- Describe an Instantaneous Description (ID) of a NPDA
- Describe a move of a NPDA using IDs
- Define how strings are accepted by a NPDA
- Show the equivalence between CFGs and NPDAs
- Define a DPDA
- Show that DPDA is not equivalent to NPDA
- Describe deterministic CFL

Self-test Questions

1. Construct an DPDA that accepts $L = \{a^n b^n : n \geq 0\}$
2. Construct an NPDA that accepts $L = \{a^{2^n} b^n : n \geq 0\}$
3. Construct an NPDA that accepts $L = \{a^n b^{2^n} : n \geq 0\}$
4. Using instantaneous description show that whether the string **aaaabb** is accepted in question 2
5. Construct an NPDA that accepts the language generated by the following CFG
$$E \rightarrow T \mid E + T \mid E - T$$
$$T \rightarrow F \mid T * F \mid T / F$$
$$F \rightarrow a \mid b \mid c \mid (E)$$
6. Construct an NPDA that accepts $L = \{ww^R : w \in \{a, b\}^+\}$
7. Construct an NPDA for the language $L = \{w \in \{a, b\}^+ : n_a(w) = n_b(w)\}$

Part II
Compiler Design
Summary

Programming machines is one of the most important components for advancement in information communication and technology. Machines solve problem as they are capable of doing several logical computations within a fraction of seconds. Machines can execute instruction when it is represented in the form of machine language which is by far different from human language. The language is defined as a set of instructions that would be executed by the machine so as to solve problems. This fascinates computer scientists to work hard and make machine programming easier and efficient and effective in the day to day human activities.

Programmer could write a non-ambiguous program (set of instructions) in human understandable form, which when executed properly on a given input would produce an output that would be desired solution to a problem. However, such languages couldn't be directly taken by the machine as it is not understandable. Such language should be translated into the machine language with the help of translators. Translator comes in different forms.

This part focus on design and implementation of compilers one of the most common translators from source code into machine language for compiled programming language. The design and implementation of programming language compiler will be introduced in this part. Theoretical aspects of language design and translation are discussed and practically demonstrated by developing a working compiler. Concepts such as lexical analysis, symbol tables, parsing, syntax directed translation, type checking, code generation, and code optimization will be introduced.

Unit One

Fundamentals of Compiler Design

Summary

Compiler is software that translator source language written in one language into the desired target language. Compilers use pre-specified grammar from a set of tokens defined using some kind of regular expressions. Compilation passes through a number of phases. The chapter will introduce fundamental concepts in design and implementation of compilers and the different phases so that students will be ready to grasp detailed concepts briefly introduced in this chapter during subsequent chapter discussion.

General Objective

The purpose of this unit is to introduce fundamental concepts of compiler to students such as definition of translators, compiler, phases of compiler design, preprocessor, linker, loader, etc

Specific Objectives

At the end of this unit, students should be able to:

- Describe what a translator is
- Describe what a compiler is
- Distinguish compiler from other translators
- Explain what the output of a compiler be for a given set of characters
- Explain the role of pre-processor, linker and loader while solving problem or during translation
- Identify the two major phases of compiler design
- Distinguish the analysis phase from the synthesis phase
- Describe the steps how a source code written using a compiled language be converted into executable machine code
- Identify and describe the three sub-phases of the analysis phase
- Explain why linear analysis is also called lexical analysis or scanner or tokenizer
- Explain why hierarchical analysis is also called syntax analysis or parser
- Identify and describe the three sub-phases of the synthesis phase

- Describe what a symbol table is in compiler design
- Describe an error handler module in compiler design
- Arrange the entire phases in the process of compiler design showing their I/O relationship
- Demonstrate the role of the above sub-phases using example

Self-test Questions

1. _____ is a program that can read a program written in high-level language and converts it into an equivalent machine language.
2. _____ is a program that translates and then executes a high-level instruction before translating the next instruction.
3. _____ is a program whose task is collecting modules of a program stored in separate files and expanding macros into source language statements.
4. _____ processes assembly language program and produces relocatable machine code.
5. _____ is a program that links together relocatable machine codes with other relocatable object and library files into the code actually runs on the machine.
6. _____ puts all executable programs into memory for execution.
7. List the various phases of a compiler
8. List the phases of a compiler that are termed as front-end of a compiler.
9. List the phases of a compiler that are termed as back-end of a compiler.
10. What are the input to and the output of lexical analysis phase?
11. What are the input to and the output of syntax analysis phase?
12. What are the input to and the output of semantic analysis phase?
13. What are the input to and the output of intermediate code generation phase?
14. What are the input to and the output of code generation phase?
15. What is the purpose of symbol table?
16. Give an example of lexical analyzer generator.
17. Give an example of parser generator.

Unit Two

Lexical Analysis

Summary

Every language has lexical element such as words, numbers, acronyms, symbols etc. In order to convey message these lexical elements should be arranged in their appropriate order. Programming language must define its lexical element before defining its set of instructions. Hence this chapter focuses on how to define lexical element and analyze them for use in subsequent phases. Moreover, before converting source code written in a compiled language, the source code lexical elements (the smallest meaning bearing units of the program) should be analyzed. This unit will provide how to do such analysis (lexical analysis)

General Objective

The purpose of this unit is to algorithmically describe how to identify the optimal sequence of tokens that exist in the source code of the program. Moreover, each of the lexical elements also called tokens will be analyzed for their lexical category, attributes (or resource requirements) for use for later analysis/synthesis phases.

Specific Objectives

At the end of this unit, students should be able to:

- Identify what the role of lexical analyzer is
- Define the terms: symbol, lexeme, token, and pattern
- Provide the set of valid symbols for one of the programming language
- Provide examples of lexemes, tokens and patterns based on the preferred language
- Describe the interaction between lexical analyzer and the parser
- Describe how lexical error be produced by the lexical analyzer
- Identify the role of regular expression for lexical analyzer
- Identify token and token property (attributes)
- Describe why lexical analyzer return token and token property after detecting a lexeme
- Describe when lexical analyzer return NULL as an attribute for a token and why
- Describe when lexical analyzer return the lexeme as an attribute for a token and why

- Describe when lexical analyzer return address of symbol table as an attribute for a token and why
- Distinguish between regular expression and regular definition
- Describe about the role of the lexical analyzer generating tools such as (Lex or Flex)
- Describe how to use Flex or Lex
- Describe the three sections of Lex or Flex source file
- Demonstrate Lex or Flex using examples

Self-test Questions

1. _____ reads the input characters of the source program, groups them into lexemes, and produces as an output a sequence of tokens for each lexeme in the source program.
2. _____ is a description of the form that lexemes of a token may take.
3. _____ is a sequence of characters in the source program that matches the pattern for a token.
4. List three additional tasks performed by a scanner other than identifying tokens.
5. List some of the tokens that exist in a particular programming language.
6. What is/are the purpose/s of attributes of a token?
7. List some of the lexical errors that can occur during scanning.
8. List some of the actions that a lexical analyzer takes in recovering from lexical errors.
9. Define a regular expression for the following tokens in C++ programming
 - a. Identifier
 - b. Keywords
 - c. Operators
 - d. Symbols
 - e. Integer numbers
 - f. Floating point numbers
 - g. Strings
 - h. Character
10. For the above tokens, what attribute each token to have if the list shows the token identity in group
11. Discuss how input buffering works with and without the use of sentinels.

12. Draw a transition diagram that recognizes the operators: +, ++, +=, --, -, -=, = and ==.
13. Distinguish regular definition and regular expression
14. Write a regular definition for integer constants in C++.
15. Give notations used in extensions of regular expressions in specifying one or more instances, zero or one instance, and character classes.
16. Discuss how a lexical analyzer system is built

Unit Three

Syntax Analyzer

Summary

Grammar is the building block to carry message through sequence of tokens. Unless the tokens are arranged in precise order, it would be difficult to decide about the instruction it carries to be performed. Hence, syntax analyzer analyzes the output of the lexical analyzer (sequence of tokens and their associated properties) so as to understand the grammar of the code. In this unit, we will focus on how to analyze the validity of the grammar of the token sequence through syntax analyzer/parser.

General Objective

The purpose of this unit is to parse token sequence using context free grammar defined for the language. Concepts such as CFG, derivation, ambiguity and ambiguity resolution, types of parsing: top-down parsing, bottom up parsing will be discussed

Specific Objectives

At the end of this unit, students should be able to:

- Define parse trees
- Describe role of Context Free Grammar for parsing
- Identify derivation in a CFG
- Distinguish left most derivation and right most derivation
- Explain how parse tree is generated from a CFG and token sequence
- Identify ambiguity in syntax analysis
- Explain how operator derived ambiguity can be resolved
- Identify what left recursion (immediate and non-immediate left recursion)
- Explain how to remove left recursion
- Explain the purpose of left factoring and how to do left factoring
- Describe top-down and bottom up parsing
- Explain what LL parsing is
- Explain the relationship between LL and top down parsing

- Explain what LR parsing is
- Explain the relationship between LR and bottom up parsing
- Define the two functions used in both top-down and bottom-up parsing: FIRST and FOLLOW
- Describe the three types of top-down parsing algorithms (recursive descent parser, recursive predictive parsing, non-recursive predictive parsing)
- Illustrate algorithmically recursive Descent parser, recursive predictive parsing, non-recursive predictive parsing with examples
- Distinguish the strength and weakness of each of the above top-down parsers
- Describe why bottom-up parser is also called shift/reduce parser
- Describe the three types of bottom-up parsing algorithms (Simple LR parser, the canonical LR parsing, look ahead LR parsing)
- Illustrate algorithmically Simple LR parser, the canonical LR parsing, look ahead LR parsing with examples
- Distinguish the strength and weakness of each of the above bottom-up parsers

Self-test Questions

1. What are the input to and the output of a parser?
2. List the two commonly used parsing methods using in compilers.
3. List the goals that an error handler should possess.
4. List four types of error recovery modes.
5. How does a compiler that implements panic mode error recovery method recovery from an error? Discuss
6. $FIRST(\alpha)$, a string of grammars symbols α , is a set of terminals that begins the strings derived from α . (TRUE / FALSE)
7. $FOLLOW(A)$, for a nonterminal A , is the set of terminals α that can appear immediately to the right of A in some sentential form. (TRUE / FALSE)
8. Compute FIRST and FOLLOW of non-terminals in the following grammar:
 $E \rightarrow TE'$

$E' \rightarrow +TE' \mid \lambda$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \lambda$

$F \rightarrow (E) \mid id$

9. List two types of top-down parsing techniques.
10. Construct a non recursive predictive parsing table for the grammar in question 8 and then parse the input `id + id $`.
11. Parse input **efef** using recursive descent parsing technique based the following grammar.
 $A \rightarrow fAeA \mid eAfA \mid \lambda$
12. What types of grammars are LL(1) grammars? What does LL(1) stand for?
13. List the procedures employed to check whether a given grammar is an LL(1) grammar or not.
14. What are handle and handle pruning in bottom-up parsing?
15. How does a shift-reduce parser works?
16. List the conflicts that occur in shift-reduce parsing and explain briefly about the conflicts.
17. List the four possible actions that a shift-reduce parser makes.
18. What does LR(K) stand for?
19. Discuss the LR parsing algorithm
20. Define an LR(0) item.
21. What is $\text{closure}(I)$, for an LR(0) item I ?
22. Define $\text{goto}(I, X)$, for an LR(0) item I and a grammar symbol X .
23. Describe the sets of items construction algorithm in SLR.
24. Describe the algorithm for constructing an SLR(1) parsing table.
25. Define an LR(1) item.
26. What is $\text{closure}(I)$, for an LR(1) item I ?
27. Define $\text{goto}(I, X)$, for an LR(1) item I and a grammar symbol X .
28. Describe the sets of items construction algorithm in LR(1).
29. Describe the algorithm for constructing an LR(1) parsing table.

Unit Four

Syntax Directed Translation

Summary

Compilers are responsible to do translation. Syntax directed translation is one of the approaches that do translation assisted by the proposed grammar. Source code will be scanned for identification of token sequence and passed to the parser. Parser succeeds if there is one or more derivations that would accepted the token sequence. Syntax directed translation refers to translating the source code while appropriate productions are selected.

General Objective

The objective of the chapter is to discuss how to approach translation of the source code while parsing. This would require understanding of the semantic of the source code while performing reduction of derivation so that translation can be made at the same time while doing reduction.

Specific Objectives

At the end of this unit, students should be able to:

- Define syntax directed translation
- Define syntax directed definition
- Describe the applications of syntax directed translation using examples
- Define attributes of grammar symbols
- Define the two types of attributes: synthesized and inherited
- Discuss the differences between synthesized and inherited attributes
- Define annotated / decorated parse tree
- Define dependency graph
- Define evaluation order and the various methods for evaluating semantic rules
- Define S-attributed grammars
- Discuss bottom up evaluation of S-attributed grammar
- Discuss top-down evaluation of S-attributed grammar
- Define L-attributed grammars

- How to define a translation scheme with inherited attribute
- How to eliminate left recursion from translation scheme

Self-test Questions

1. _____ is a generalization of the CFG in which each grammar symbol has an associated set of attributes (synthesized and inherited) and rules.
2. List the two types of attributes in syntax directed definition.
3. What is an annotated parse tree?
4. Given the following syntax directed definition, draw annotated parse tree for 1101.11

Syntax Rules	Semantic Rules
$N \rightarrow L_1.L_2$	$N.v = L_1.v + L_2.v / (2^{L_2.l})$
$L_1 \rightarrow L_2 B$	$L_1.v = 2 * L_2.v + B.v$
	$L_1.l = L_2.l + 1$
$L \rightarrow B$	$L.v = B.v$
	$L.l = 1$
$B \rightarrow 0$	$B.v = 0$
$B \rightarrow 1$	$B.v = 1$

5. What types of attributes are v, l, and s in question 4? (synthesized / inherited)
6. Draw a dependency graph for question 4.
7. Write at least two topological sort for question 4.
8. What is an S-attributed grammar?
9. What is an L-attributed grammar?
10. What is the difference between attributes in an L-attributed grammar and inherited attributes?
11. Give an example of S-attributed grammar.
12. Give an example of L-attributed grammar.

Unit Five

Type checking

Summary

Some programming are strongly typed (require all its identifiers to be declared before use and implicitly type casting is prohibited), some are weakly typed (require all its identifiers to be declared before use but implicitly or explicit type casting is possible among compatible types) and others are un-typed (doesn't require identifiers to be declared and they can play any type role at any point in the program). Type checking is required for strongly or weakly typed languages to conform the design policy.

General Objective

The objective of this chapter is to discuss the issue of type checking and detect invalid operation from the perspective of object type so that corrective measure can be made at early stage.

Specific Objectives

At the end of this unit, students should be able to:

- Define what type checking is
- Describe the various types of static checks: type checks, flow-of-control checks, name-related checks, uniqueness checks
- Define type expressions
- Define type constructors
- Show how type constructors can be used in defining type expressions
- Define type systems
- Describe error recovery in type checking
- Show assigning types to the various parts of a program using syntax directed definition
- Describe equivalence of types
- Describe structural equivalence of types
- Describe name equivalence of types
- Describe type conversions

Self-test Questions

1. List some of the activities that a compiler performs during type checking.
2. What does uniqueness check mean?
3. List some of the type expressions in C++.
4. List some of the type constructors in C++.
5. Describe the difference between structural equivalence and name equivalence in type expressions.
6. Given the following syntax directed definition for assigning types for program parts, draw an annotated parse tree for **a mod b** assuming that a and b have been declared integers.

E → literal	{E.type := char}
E → num	{E.type := integer}
E → id	{E.type := lookup (id.entry)}
E → E1 mod E2	{E.type := if E1.type = integer and E2.type := integer then integer else type_error}
E → E1[E2]	{E.type := if E2.type = integer and E1.type := array(s, t) then t else type_error}
E → ^E1	{E.type := if E2.type = pointer(t) then t else type_error}

Unit Six

Intermediate Code Generation

Summary

Usually compilers don't generate the machine code or object code on the fly. It should pass through phases such as intermediate code generation. Intermediate codes are codes that split statements into a set of smaller instructions that could be directly mapped into the desired object code. This permits to easily generate the final code and to undertake code optimization

General Objective

The main objective of this chapter is to generate an equivalent code as the source code which can be easily mapped into the desired object codes.

Specific Objectives

At the end of this unit, students should be able to:

- Define intermediate code
- Describe the use of generating machine-independent intermediate code
- List various types of intermediate languages: syntax trees, postfix notation, three – address code
- List the various types of three address statements: binary assignment, unary assignment, copy statement, unconditional jump, conditional jump, function call, indexed arguments, and address and pointer assignments
- Use syntax directed translation to generate three-address code statements for: declarations, assignment statements, and addressing array elements
- Define backpatching

Self-test Questions

1. What are the uses of generating an intermediate code?
2. List three intermediate languages used in intermediate code generation.
3. Give the format of three-address code for function call.
4. Give the format of three-address code for conditional jump.

5. Describe the uses of the functions: **newtemp**, **newlabel**, and **gen** in generating three-address code statement using syntax directed translation.
6. Given the following syntax directed definition, generate three-address code statements for the assignment statement: $a := x + y * z$.

Production Semantic Rules

$S \rightarrow id := E$	$S.code := E.code \parallel gen(id.lexeme, :=, E.place)$
$E \rightarrow E_1 + E_2$	$E.place := newtemp;$ $E.code := E_1.code \parallel E_2.code \parallel gen(E.place, ':=', E_1.place, '+', E_2.place)$
$E \rightarrow E_1 * E_2$	$E.place := newtemp;$ $E.code := E_1.code \parallel E_2.code \parallel gen(E.place, ':=', E_1.place, '*', E_2.place)$
$E \rightarrow - E_1$	$E.place := newtemp;$ $E.code := E_1.code \parallel gen(E.place, ':= uminus ', E_1.place)$
$E \rightarrow (E_1)$	$E.place := newtemp;$ $E.code := E_1.code$
$E \rightarrow id$	$E.place := id.lexeme;$ $E.code := '' /* empty code */$

Unit Seven

Code generation

Summary

Code generation is the final phase in compiler implementation which generates the object code from the intermediate codes. Code generator may be undertaken before or after code optimization.

General Objective

The objective of this chapter is to discuss how code can be generated from the intermediate code.

Specific Objectives

At the end of this unit, students should be able to:

- Define code generation
- Describe the properties that the generated code should possess
- Describe the issues in code generation including: input, output, memory management, instruction selection, and register allocation
- Describe how the simple code generation algorithm works

Self-Test Questions

1. What are the input to and the output of a code generator?
2. What properties are required of the generated code?
3. Give an example on how selection of instructions affects the efficiency of a generated code.
4. Describe the differences between register allocation and register assignment.
5. Given the following three-address statements for the assignment statement $d := (a - b) + (a - c) + (a - c)$, describe how the simple code generation algorithm works assuming that d is live.

$t := a - b$

$u := a - c$

$v := t + u$

$d := v + u$

Study Guide

For

Fundamentals of Programming

Department of Computer Science
Faculty of Informatics



ቅድስት ማርያም ዩኒቨርሲቲ
St. Mary's University, Ethiopia

Prepared by
Sebsibe Hailemariam (PhD)

January 2012

Summary

Fundamental of programming is a course offered for junior students at the Department of Computer Science and other computing field of study. The course can be seen as a corner stone to Computer Science students as it provides diverse insight into the field of the Science. In the first place, this course indicates what students are going to do in the future after completion of their course. Secondly, this course gives broader view on how computer solve users problem which is also one of the intended profile of students to attain upon completion of the program. Moreover, the course allows students to see briefly how computer function in order to do some task.

This guideline is not a reference that students should use during the period where the course fundamental of programming is offered. Rather it can be used for the students as a checklist to know the expectations that students are expected to know and what they already know. Hence, this guideline clearly shows for the students' existence of any gap among the expected knowledge, skill and attitude to be achieved at the end of the course.

In addition, this guide line is relevant not only to students but also for instructors that teaches this course. Instructors should use this guide line as a check list to see their periodic status while teaching the course and plan a head how to cover the important topics identified that instructors are expected to teach students and transfer knowledge and skill to their students.

This guideline is an important starting point to someone who wants to learn programming with independent study. Finally, the guide line is an important asset to the Department to give a direction to instructors what needs to be covered while he/she is teaching the course.

Unit one

Computer Systems and Programming Language

Summary

Programming is an art of instructing what the computer hardware should do in order to fulfill the needs and requirements of users. Programmers can write efficient and high quality programs if they know important elements of the computer hardware and the software that run under the computer system for various reasons. This chapter is intended to give a direction to students what they should know about the computer hardware and software to be successful programmer.

General Objective

The purpose of this unit is to revise basic computer system components and the role of the components in solving problem by programming and to develop algorithmic thinking so that students can translate a given problem into algorithm that can be implemented using computer programming language.

Specific Objectives

At the end of this unit, students should be able to:

- Describe the different hardware components of the computer system and their functionalities
- Distinguish Input unit, output units, RAM, secondary storage devices, CPU, Arithmetic Logical Units, Control Unit, Register, Cache Memory, communication bus, (Address Bus, Data Bus, and other communication systems that links different components of the computer hardware)
- Identify the role of these hardware components
- Describe computer software
- Identify the classification scheme of software
- Distinguish computer software from computer hardware
- Explain what an Operating System is
- Identify the role of a computer operating system
- Distinguish the different types of operating systems
- Describe what language software is
- Distinguish the different types of language software

- Identify the different generations of language software
- Explain what is application software is
- Describe how to solve problem using Computer
- Describe the steps to solve problem using computer
- Distinguish the different phases in solving problem using computer (Understanding the problem, designing the algorithm, implement the algorithm, debugging and testing the program, deploying the solution)
- Describe how to develop algorithm using flow chart or pseudo-code
- Describe the translation procedure of source code into machine understandable code
- Describe how to translate source code into machine code for different types of source codes (Assembly language code, compiled language code or interpreted language codes)
- Distinguish the difference among the different translators (Assembler, Interpreter and Compiler)
- Identify some sample programming languages which are compiled or interpreted types
- Describe the steps from writing a code in compiled language until execution
- Describe the concepts of pre-processing, compiling, linking, loading, executing
- Describe the role of the operating system while the loader loads an executable program
- Distinguish code segment and data segment in the RAM for a process?
- Describe the three types of programming error?
- Identify the basic characteristic of syntax (compiled type) error?
- Identify the basic characteristic of logical error?
- Identify the basic characteristic of the runtime error?

Self-test Questions without answer key

Attempt all of these questions and make sure that you answer them correctly.

1. Explain the role of the basic components of a computer hardware: RAM , Register, Cache memory, Address bus, Data bus, ALU, Control unit, Secondary storage device, etc
2. What is clock cycle of a CPU? How many clock cycle an instruction will take?

St. Mary's University
Faculty of Informatics

3. Compare & contrast speed of fetching data by the CPU from different data source location such as Register, Cache memory (L_1, L_2 & L_3), RAM, Hard disk, Flash disk, CD, Magnetic tape.
4. Explain clearly about pre-processing, Compiling, Linking, Loading and Executing.
5. Explain why OS assign code segment, Data Segment, Heap Segment & Stack Segment to each program loaded to be executed by the CPU.
6. Write a flowchart that accept a number and check whether the number is even or odd
7. Write a flowchart that search a given searchKey value is in the list of values
8. Write a flow chart that checks whether a given number is prime or not. A prime number is a number that has only two distinct factors (1 and the number itself)
9. Write a flowchart that search the smallest prime number greater than or equals to N.
10. Write a flowchart that searches the N^{th} prime number.
11. Discuss the different types of pre-processing directives.
12. It has been decided that a bonus of 12% is to be given for each employee in an organization. It was also agreed that if an employee has worked for more than 13 years, s/he is to receive an additional amount of Birr 350.00. Draw a flow chart that describes how the bonus for an employee is calculated given his/her current salary.
13. Write a flowchart that display sequence of n positive integers starting from a user specified number m
14. Write a flowchart that convert a mark for a course to its corresponding letter-grade using the following scale

<u>Mark</u>	<u>Grade</u>	<u>Mark</u>	<u>Grade</u>
≥ 90	A+	≥ 55	C+
≥ 80	A	≥ 45	C
≥ 75	B+	≥ 30	D
≥ 60	B	< 30	F
15. Write a flowchart that compute and display the sum of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.
16. Write a flowchart that compute and display the product of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.

17. Write a flowchart that accept two numbers and an operator and compute and display the result of the expression (be careful to handle division by zero)
18. Write a flowchart that accept N integers and display them in reverse order
19. Write flowchart that computes the number of days in a given month and year of a Gregorian calendar.
20. Write a flowchart that compute and display the next date of a specific date in European calendar accepted from the user. You must consider leap year condition along with other conditions.
21. Write a flowchart that accept a list of integers different from zero and terminate accepting the integers when the input is zero.
22. Modify the above flowchart so that it would display the number of positive and negative integers
23. Write a flowchart that calculate income tax of an employee given the total monthly salary from the keyboard and following the following personal tax rule of Ethiopia
 - a. All salary amounts less than or equal to 150 Birr are tax free.
 - b. Salary amounts between 151 to 650 birr will be deducted 10% of the salary - 15 birr
 - c. Salary amounts between 651 to 1400 birr will be deducted 15% of the salary - 47.50 birr
 - d. Salary amounts between 1401 to 2350 birr will be deducted 20% of the salary - 117.50 birr
 - e. Salary amounts between 2351 to 3550 birr will be deducted 25% of the salary - 235 birr
 - f. Salary amounts between 3551 to 5000 birr will be deducted 30% of the salary - 412.50 birr
 - g. Salary amounts above 5000 birr will be deducted 35% of the salary - 662 birr
24. Write a flowchart that search and display the maximum and the minimum of a list of numbers entered by the user. Where the size of the list is to be entered by the user.

25. Write a flowchart that calculate and display the factorial of a number n where n is a positive number to be entered by the user.

Questions with answer key

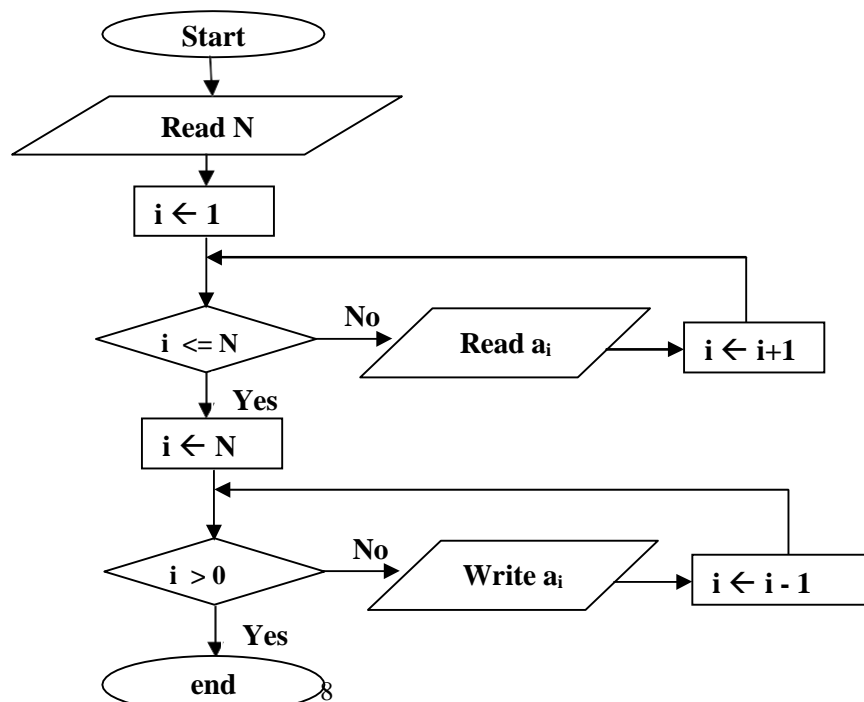
1. A code (Source program) written in C++ cannot be executed directly by the machine (Computer). Why? What needs to be done?
2. Give some list of C++ compilers
3. Explain compile time/syntax error, Logical & Run time errors
4. Design a flow chart to accept N numbers and display them in reverse order
5. Write a flowchart that accept a list of integers different from zero and terminate when the input is zero.
6. Write a flowchart that calculate income tax of an employee given the total monthly salary from the keyboard following the following personal tax rule of Ethiopia and display the tax on the screen
 - a. All salary amounts less than or equal to 150 Birr are tax free.
 - b. Salary amounts between 151 to 650 birr will be deducted 10% of the salary - 15 birr
 - c. Salary amounts between 651 to 1400 birr will be deducted 15% of the salary - 47.50 birr
 - d. Salary amounts between 1401 to 2350 birr will be deducted 20% of the salary - 117.50 birr
 - e. Salary amounts between 2351 to 3550 birr will be deducted 25% of the salary - 235 birr
 - f. Salary amounts between 3551 to 5000 birr will be deducted 30% of the salary - 412.50 birr
 - g. Salary amounts above 5000 birr will be deducted 35% of the salary - 662 birr

Answer key

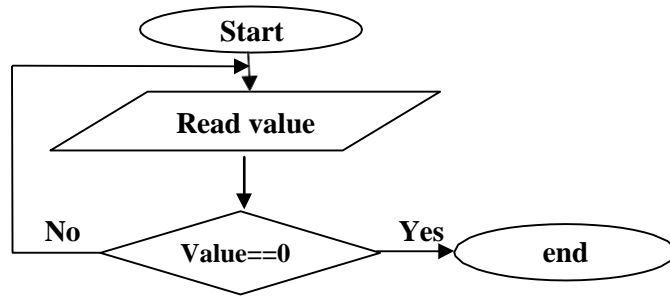
1. A code (Source program) written in C++ cannot be executed directly by the machine as the C++ code is not directly understandable by the machine. Machine code is codes written in zeros and ones where as C++ codes are written using sequences of ASCII characters. The C++ code needs to undergo a number of stages to be executed by the machine. First, the preprocessor must handle the source code to handle all preprocessing directives. The output of the preprocessor will be given to the compiler so that all instructions will be translated into the equivalent object code. The object code should be linked with other required object codes (can be pre-compiled library objected codes or user defined library object codes). The output the linker can be executed if the program is loaded into the RAM as the RAM is the working memory of the computer.
2. The following are some of the C++ compilers available for use in writing C++ program

Borland C++	GCC
Turbo C++	C++ Compiler professional
Dev-C++	Visual C++
3. Syntax/compile time error is an error caused as a result of wrong lexical or syntactic elements of the program construct which cannot be translated by the compiler. Logical error is an error due to the computational logic of the program that results wrong output which may mislead users decision. Runtime error is an error that result abnormal termination of program execution.

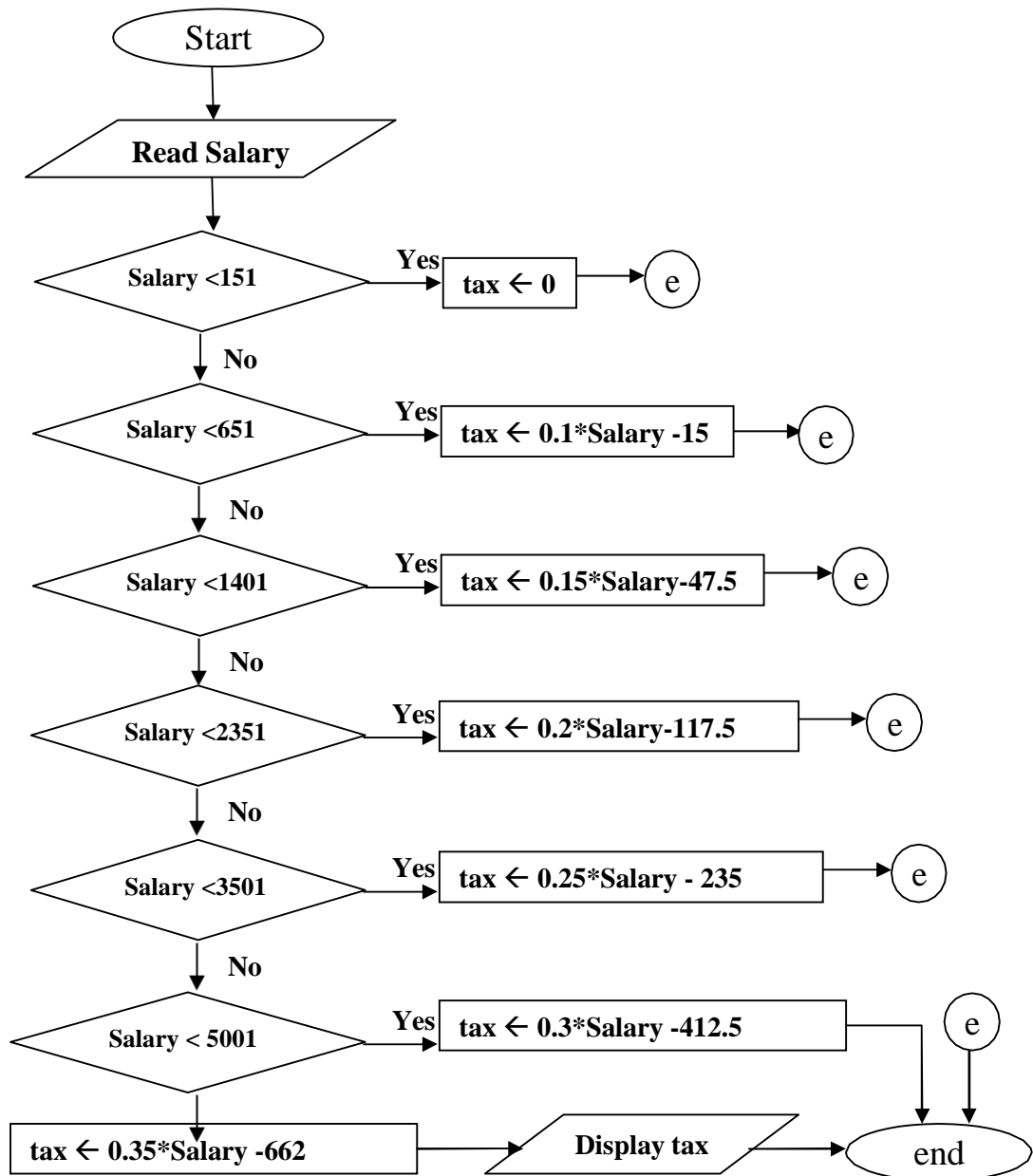
4.



5.



6.



Unit Two

Basics of Programming Language

Summary

Program is a language written using selected programming language. Programming language as a language has its own lexical elements and statements constructed from the lexical elements. The statements of a program written in a selected programming language follow its own grammar. This is exactly the same as the natural language that we use in day to day communication such as Amharic, English, Afaan Oromo, Tigregna. These languages has words, numbers, punctuation marks, acronyms as lexical elements and grammar that define legal sequence of such lexical elements to form valid statement. Lexical terms has different role such as subject maker, object maker, linking clauses, etc and the same is true in programming language. A program is hence a sequence of statements that potentially instruct the computer hardware what to do to solve a problem.

General Objective

The purpose of this unit is to introduce student with general structure of a program in a programming language; simple grammar of basic statements and expressions; lexical elements and issues associated with them for use in statements/expressions of the selected programming language (C++ is chosen for this guideline).

Specific Objectives

At the end of this unit, students should be able to:

- Describe the structure of the C++ program
- Describe preprocessing directives
- Identify the most common preprocessing directives
- Use preprocessing directives
- Describe global declaration statements (Variable identifier declaration, Constant identifier declaration, Function declaration, External function declaration)
- Explain how many functions a C++ program can have
- Identify the elements of a function definition and implementation section
- Describe function return type, function name, function argument

- Distinguish local versus global variables
- Identify constant identifier
- Explain constant identifier declaration statement
- Identify what an identifier is (one of the lexical element)
- Define an identifier
- Identify keywords in C++ programming
- List examples of keywords in C++ and their purpose
- List some examples of non-keyword identifiers
- Discuss the purpose of identifier (As a function name, As a variable identifier, As a constant identifier, As a keyword, As user defined data type or class name)
- Describe declaration statement
- Identify the syntax of declaration statement
- Identify data type in declaration statement
- Identify the purpose of data type to compilers
- Show how to declare variable identifier with/without initialization
- Show how to declare constant identifier
- Describe expressions in C++ program
- Define an expression
- Identify the ingredients of an expression
- Identify the valid operands in an expression (Function calls, Variable identifiers, Constant identifiers, Literal constants)
- Identify the valid operators in an expression and their semantics (Arithmetic operators, Relational operators, Logical operators, Increment operator, Decrement operator, Bitwise operator, Conditional operator (?:), Comma operator, Sign operator (+ and -), Assignment operator, Parenthesis)
- Identify the unary, binary and tertiary operators in C++ programming language
- Describe operator precedence and show the operator precedence rule in C++
- Describe association rule of operators and state the association rule of the different operators
- Identify and use special characters represented using escape sequence

- Describe assignment statements
- Identify the syntax for assignment statement
- Identify how assignment statement is evaluated
- Describe output statements
- Define an output statement
- Identify the syntax for output statement
- Show how to output string, expression value, etc
- Explain how output statement is evaluated
- Show some example of simple output statements
- Describe input statement
- Define input statement
- Identify the syntax for the input?
- Show sample example of simple input statement
- Describe the syntax for the main function definition
- Explain why we need data type before the function name (Is it optional or not)
- Identify the possible data types that can be used for the main function
- Explain whether the main function name is case sensitive or not
- Defend that the main function is compulsory for every C++ programs
- Explain the purpose of the parenthesis “()” for the main function
- Describe what can be placed inside the parenthesis?
- Describe return statement inside a main function
- Identify why we need return statement in the main function
- Construct simple program consists of declaration statement, assignment statement, input and output statements)

Self-test Questions

1. Where the preprocessor finds the files needs to be included and defined using the #include preprocessor directive
2. Give at least four invalid identifier names each having different reasons for being invalid
3. Write three possible ways of writing a C++ program main function definition
4. Give a reason why the following sequence of characters are not identifiers

**St. Mary's University
Faculty of Informatics**

- a. void
 - b. 1_test
 - c. Phone number
 - d. AmountIn%
 - e. _Salary@eth
 - f. asm
5. Given a declaration statement, “**double x = 7.0; char ch; float f; short int age;** ”, how much memory space would be reserved for **X, ch, f, and age**
6. Show the order of evaluation for the following expressions
- a. $Z = a \&\& b \parallel !c$ (assume $a = 10$; $b = 0$; $c = -2$)
 - b. $4 + 5 * 2 / 4 * 0.5$
7. Why the variable to the left of an assignment statement is a variable identifier?
8. Write a simple output statement that output the message “**Hello world**” on the screen
9. Write a simple output statement that output the message the value of the expression **(4+5*2 /4*0.5)** and the expression **(10 + Y * (Z - 5))** separated by **tab** character
10. Write a simple input statement used to read value for the three variable identifier X, Y and Z
11. A programmer wrote an input statement **cin>>N;** where N is a declared constant identifier. The compiler refused to compile this line of statement. Why do you think is that?
12. Write a program that display the message “Hello World!!” on the screen
13. Write a program that display the number from 1 to 4 separated by tab character
14. Write a program that display the number from 1 to 4 in a separate line using only one output statement
15. Write a program that reads one integer variable, one character and one double and display them in reverse order

Questions with answer key

- 1. Write a “*preprocessor directive statement*” that define MAX to be 1000
- 2. Write a preprocessor directive that includes **string.h**
- 3. Write a declaration statement that declares a float variable identifier called mass with initial value of zero.
- 4. Write a declaration statement that declares a constant identifier called gravity with value 9.8

5. Show the order of evaluation for the expression $X = 10.5 + Y * (Z - 5)$ assuming the following value ($X = 5$; $Y = 10$; $Z = 2.5$). Note that the data type can be guessed from the current value given
6. Write a program that accept three floating point numbers and display their sum

Answer key

1. ***#define MAX 100***
2. ***#include<string.h>***
3. ***float mass=0;***
4. ***const float gravity 9.8;***
5. $X = 10 + Y * (Z - 5)$
 - a. First evaluate the subtraction operator that results $X = 10.5 + Y * 2.5$
 - b. Next evaluate the multiplication operator that results $X = 10.5 + 25.0$
 - c. Next evaluate the addition operator that results $X = 35.5$
 - d. Finally the value of the expression which is 35.5 will be assigned to X. However, as X is an integer, 35.5 will be first truncated into 35 and 35 is assigned to X.

```
6.  
#include<iostream.h>  
int main()  
{  
  
    float f1,f2,f3;  
    cout<<"Enter the three float numbers ";  
    cin>>f1>>f2>>f3;  
    cout<<"The sum of the three float number = "<<f1+f2+f3<<endl;  
    return 0;  
}
```

Unit Three

Control Statements

Summary

Solving problem by computer needs computation of finite set of instructions step by step in a well-controlled fashion. Usually instructions are placed in a program ordered by line number and from left to right if more than one instruction is placed in one line. However, the execution plan to solve a given problem may require executing the current instruction to determine the next instruction to be executed. Hence, the next instruction to be executed cannot be determined just by looking the current instruction line number. Instructions that determine what to be the next instruction to be executed after they get executed are called control instructions/statements. If the current instruction is not a control statement, then the next instruction that would be executed is the one which is located next to the current statement in their logical ordering (line number and left to right in a line). This unit focuses on providing students a guide about control flow and control statements as well as their computations strength.

General Objective

The purpose of this unit is to introduce student with control statements and structures appropriate for the different possible control statements.

Specific Objectives

At the end of this unit, students should be able to:

- Describe control flow in programming
- Identify the purpose of an instruction pointer or program counter register
- Identify the difference between sequential execution and non-sequential execution
- Identify how the value of instruction pointer or program counter register value changes
- Identify the difference between conditional and unconditional control statements
- Identify the two basic types of conditional control statements: branching and looping
- Describe branching/selection control statements
- Identify the difference between sequential statement execution and branching control statement execution
- Identify the syntax of an if control statement

St. Mary's University
Faculty of Informatics

- Identify the valid and invalid conditional expressions that can be used in an “if statements
- List all alternative syntaxes for the if statement
- Apply the most appropriate if statements in program writing
- Distinguish block statements and simple statement in program
- Identify the section of the code that the if statement applies to
- Identify the syntax of an switch control statement
- Identify the valid and invalid switch statement expressions
- List the method how to use case statement
- Know the purpose of break statement in case statements
- Apply switch statements in program writing as appropriate
- Identify the section of the code that the switch statement applies to
- Formulate what kind of flowchart can be used to represent if statements and switch statements
- Describe iterative/looping control statement
- Identify the difference between branching control statement execution and looping control statement execution
- Identify the different looping constructs in C++
- Identify counter controlled loop and sentinel based loop
- Identify the C++ control statements appropriate for counter controlled loop
- Identify the C++ control statements appropriate for sentinel based loop
- Identify the syntax of a for loop, while loop and do..while loop control statement in C++
- Identify when a for loop control statement is more appropriate that the others
- Identify when a while loop control statement is more appropriate that the others
- Identify when a do..while loop control statement is more appropriate that the
- Identify the valid and invalid conditional expressions that can be used in these looping control statements
- Apply the most appropriate looping control statements in program writing
- Identify the section of the code that the loop statement applies to
- Formulate what kind of flowchart can be used to represent a for loop, while loop and do..while loop statements in C++

- Identify the role of the break statement in a loop construct
- Identify the role of the continue statement in a loop construct
- Identify the role of the return statement in a loop construct

Self-test Questions

1. Write a program that accept a number and return the type of the number (“even or odd”)
2. Write a flowchart that search the smallest prime number greater than or equals to N.
3. Write a program that display sequence of n positive integers starting from a user specified number m
4. Write a program that convert a mark for a course to its corresponding letter-grade using the following scale

<u>Mark</u>	<u>Grade</u>	<u>Mark</u>	<u>Grade</u>
>=90	A+	>=55	C+
>=80	A	>=45	C
>=75	B+	>=30	D
>=60	B	<30	F

5. Write a program that compute and display the sum of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.
6. Write a program that accept two numbers and an operator and compute and display the result of the expression (be careful to handle division by zero)
7. Write a program that computes the number of days in a given month and year of a Gregorian calendar.
8. Write a program that compute and display the next date of a specific date in European calendar accepted from the user. You must consider leap year condition along with other conditions.
9. Write a program that search and display the maximum and the minimum of a list of numbers entered by the user. Where the size of the list is to be entered by the user.
10. Write a program that calculate and display the factorial of a number n where n is a positive number to be entered by the user.
11. Design a flow chart to accept N numbers and display them in reverse order

Questions with answer key

1. Write a program that checks whether a given number is prime or not. A prime number is a number that has only two distinct factors (1 and the number itself)
2. Write a program that compute and display the product of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.
3. Write a program that accept a list of integers different from zero and terminate when the input is zero. The program finally display the number of positive and negative integers
4. Write a program that calculate income tax of an employee given the total monthly salary from the keyboard following the following personal tax rule of Ethiopia and display the tax on the screen
 - a. All salary amounts less than or equal to 150 Birr are tax free.
 - b. Salary amounts between 151 to 650 birr will be deducted 10% of the salary – 15 birr
 - c. Salary amounts between 651 to 1400 birr will be deducted 15% of the salary – 47.50 birr
 - d. Salary amounts between 1401 to 2350 birr will be deducted 20% of the salary – 117.50 birr
 - e. Salary amounts between 2351 to 3550 birr will be deducted 25% of the salary – 235 birr
 - f. Salary amounts between 3551 to 5000 birr will be deducted 30% of the salary – 412.50 birr
 - g. Salary amounts above 5000 birr will be deducted 35% of the salary – 662 birr

Answer key

```
1.
#include<iostream.h>
int main(){
    int N;
    cout<<"Enter the value of N - -> "; cin>>N;
    if(N < 2) {cout<<N<<" is not prime "<<endl; return 0; }
    if(N == 2) {cout<<N<<" is prime\n"; return 0; }
    else{
        for(int i=2; i <N; i++){
            if(N % i == 0){cout<<N<<" is not prime\n"; return 0; }
        }
        cout<<N<<" is prime \n"; return 0;
    }
}
```

```
2.
#include<iostream.h>
int main(){
    int product=1, start, end;
    cout<<"Enter the starting and ending number - -> "; cin>>start>>end;
    for(int i=start; i <=end; i++)product *=i;

    cout<<"The product of integers from "<<start<<" to "<<end<<" = "<<product<<endl;
    return 0;
}
```

3.

```
#include<iostream.h>

int main(){
    int countPositive=0, countNegative=0, value;
    do{
        cout<<"enter a number 0 to exit "; cin>>value;
        if(value > 0) countPositive++;
        else if(value < 0) countNegative ++;
    }while(value !=0);
    Cout<<"You entered "<<countPositive << " positive numbers and ";
    Cout<<countNegative<<" negative numbers\n";
    return 0;
}
```

4.

```
#include<iostream.h>

int main(){
    float grossSalary, incomeTax;
    cout<<"Please enter the gross salary "; cin>> grossSalary;
    if(grossSalary <= 150) incomeTax=0;
    else if(grossSalary <= 650) incomeTax= grossSalary*0.1 – 15.0;
    else if(grossSalary <= 1400) incomeTax= grossSalary*0.15 – 47.50;
    else if(grossSalary <= 2355) incomeTax= grossSalary*0.2 – 117.50;
    else if(grossSalary <= 3550) incomeTax= grossSalary*0.25 – 235.0;
    else if(grossSalary <= 5000) incomeTax= grossSalary*0.3 – 412.50;
    else incomeTax= grossSalary*0.35 – 662.0;
    cout<<"The employee income tax = "<<incomeTax<<endl;
    return 0;
}
```


Unit Four

Arrays, Strings and Pointers in C++

Summary

Solving problem by computer needs computation of finite set of instructions step by step in a well-controlled fashion. This computation is made to process input data and transform the data into output information. The data which is the primary source of the information must be stored so that computation can be made systematically and efficiently. Usually the data needs to be manipulated is large in size and complex in structure. Array, string and pointers are structures in programming used to represent data for use in computational processing.

General Objective

The purpose of this unit is to introduce student with the array, string and pointer constructs that are vital to represent large and complex data for efficient computational algorithm.

Specific Objectives

At the end of this unit, students should be able to:

- To explain what an array is and how array objects are structured in memory
- To explain why array elements must have the same data type
- To explain why array elements are arranged in contiguous memory space
- Identify how array elements are referred using the array name and how the exact location of the array be calculated
- Describe how array identifier is declared and initialized
- Identify similarities and differences between array element referred by the array name and its index and simple variable identifier with the same type as the array elements
- Explain how array can be passed as an argument to a function
- Identify the difference between array of character and string
- Explain how to use array and string for representing collection of data
- Explain how to initialize array of character with string value
- Identify how array elements are referred using the array name
- Identify some of the most important string manipulation functions and how to use them

- Explain how to convert string into integer, long integer or floating point number
- Distinguish 1-dimensional and multi-dimensional array
- Identify the syntax to declare multi-dimensional array identifier
- Illustrate 2-D array as array of elements where each elements are 1-D array
- Illustrate 3-D array as array of elements where each elements are 2-D array
- Apply multi-dimensional array to represent, access and process complex data
- Explain what a pointer and pointer variables are
- Distinguish the difference between address and value of a variable identifier
- Identify the syntax to declare pointer variable
- Explain why pointer variable declaration require type specification
- Explain what is a NULL pointer value
- Identify how to assign pointer variable a value which is address of compatible variable identifier, element of an array identifier, value of array identifier
- Explain the semantics/meaning of arithmetic on pointer variable identifier
- Identify how to access value of an object using the array variable value which store the address of the object
- Identify how to allocate memory dynamically to the pointer variable
- Explain how to manage dynamically allocated memory using pointer variable
- Identify how to de-allocate memory allocated dynamically and pointed by the pointer variable
- Explain what an array of pointer is
- Explain how to reserve memory, use the memory, and de-allocate the memory assigned for an array of pointer
- Explain about pointer of pointer variable identifier and its declaration
- Identify how to assign memory location dynamically to pointer of pointer variable identifier
- Identify how to de-allocate memory location dynamically assigned to pointer of pointer variable identifier
- Identify why one needs a pointer variable of type void *

Self-test Questions

St. Mary's University
Faculty of Informatics

1. Write a program that stores temperatures for 4 weeks to an array from the user, and identify the week with the highest average temperature (1-week is 7 days).
2. Rewrite question number 2 and sort them in ascending order and display the numbers before and after sorting.
3. Rewrite question number 2 and sort them in descending order and display the numbers before and after sorting.
4. Write a program that counts the number of times an item appears among the elements of a list of integers stored in an array from the keyboard and displays that count as the frequency of the number in the list.
5. Write a program that will reverse the first half elements of an array of integers with the second half elements symmetrically, without the need to declare another array. E.g. An array having the elements {a,b,c,d,e,f} when reversed, it will be {d,e,f,a,b,c}.
6. Write a program that accept N integers and display them in reverse order. Use pointers to avoid over estimation or under estimation of memory reservation)
7. Write a program that tracks the grades for 5 classes, each having 10 students. Input the data through keyboard. Print the table in its native row and column format.
8. Modify the above question assuming that the number of students in each class is not known. (Hint: use pointer of array)
9. Write a program that merges two sorted integer arrays in a third array. The merged array should be sorted too.
10. Write a program that stores and prints the numbers from 1 to 21 in a 3-by-7 table.
11. Write a program that accepts a square matrix ($N \times N$ dimension where N is unknown) and displays the summation of:
 - a. the rows separately
 - b. the columns separately and
 - c. the two diagonals(Use pointer of pointers to properly manage the required memory space to hold the matrix)
10. Write a program that checks whether a word is palindrome (read the same forwards and backwards) or not.

11. Write and test a program that performs perfect shuffle on an array of integers. A perfect shuffle is a sequence obtained by interleaving its first half with the second half, always moving the middle card to the front. For example, the perfect shuffle of {1,2,3,4,5,6,7,8,9,10} is {6,1,7,2,8,3,9,4,10,5}
12. Write a program that reads two 4x4 square matrices, A and B, and then displays
 - a. $A+B$
 - b. $A-B$ and
 - c. $A*B$
13. Extend the above question so that dimension is $N \times N$ and apply concept of pointer
14. Write and test a program that transposes a square matrix.
15. Write a program to find the number of times that a given word occurs in a sentence

Interaction with the program might look like:

The word is "the".

The sentence is "the cat sat on the mat".

The word occurs 2 times.

16. SMUC plan to have a system that keep daily track of students ID that enter into the campus. A student ID is 14 characters long. Initially N (say $N=1500$) byte long buffer of character is prepared to keep $N/15$ (100) students ID. The user keeps entering students' ID when they enter into the campus. However, after a while, the buffer may be full and require creating a new buffer of size $2N$ and transferring the old buffer data into the new buffer and the old buffer size should be reused. Again if the new buffer get full, we should create $3N$ sized buffer, $4N$ sided buffer and so on. Write the program that successfully implements this idea.

Questions with answer key

1. Write a program that Search a searchKey from an array of N integers constructed by getting its value from a file (call the function **getData** with the array and its MAX size as its arguments. The function will read the data and store into the array and return the number of integers stored if successful; 0 if the data doesn't exist or size exceeds the specified size or any other problem exist). The program should return the index of the array if it founds the searchKey or -1 to mean, it doesn't exist.
2. Write a program that accept N integers from the user and print the maximum, the minimum, average and standard deviation of the data values.
3. Write a program that stores daily sales price data of a company XYZ which sells computer spare parts. XYZ operate every day including Sunday for the last 2 years. There are also different types of items available in the Company shops. You are asked to define appropriate pointer that can store the amount of birr sold for a given item and a given day. (Note: item and day can be easily represented by integer). For example the first item and in the second day can be represented by the integer 0, 1. Finally display the total sales of each item in all days.

Answer key

1.

```
#include<iostream.h>
#define MAX 1000
int main(){
    int data[MAX], key, size;
    size = getData(data, MAX);
    if(size == 0){
        cout<<"unable to get the data "<<endl;
        return 1;
    }
    cout<<"Enter your integer to search "; cin>>key;
    int i=0;
    while(data[i++] !=key && size > i);
    if(i < size) cout<<"I found the ket at index "<<i<<endl;
    else cout<<"The key doesn't exist in the list"<<endl;
    return 0;
}
```

2.

```
#include<iostream.h>
int main(){
    int *data, min,max,size, sum=0;
    float average, stdv=0;
    cout<<"How many integers you need to store ==>"; cin>>size;
    data = new int[size];
    if(size == 0) return 0;
    cout<<"Enter data[0] = "; cin>>data[0];
    max = min = data[0];
    for(int i=1; i < size; i++){
        cout<<"Enter data["<<i<<"] = "; cin>>data[i];
        sum +=data[i];
        if(min > data[i]) min = data[i];
        if(max < data[i]) max = data[i];
    }
    average = float(sum)/float(size);
    for(int i=0; i < size; i++)
        stdv +=(data[i]- average)*(data[i]- average);
    stdv = stdv/size;
    cout<<"Distribution of the data\n";
    cout<<"Min Value      \t\t"<<min<<endl;
    cout<<"Max Value       \t\t"<<max<<endl;
    cout<<"Average Value\t\t"<<Average<<endl;
    cout<<"Standard Deviation\t"<<stdv<<endl;
    return 0;
}
```

```
3.
#include<iostream.h>
int main(){
    float **data;
    int rows, cols;
    cout<<"How many days of data you need to store==>"; cin>>rows;
    cout<<"How many items of data you need to store==>"; cin>>cols;
    data = new (float *) [rows];
    for(int i=0; i<rows; i++){
        data[i] = new float[cols];
        cout<<"enter the data for day="<<i+1;
        for(int j=0; j<cols; j++){
            cout<<" for item="<<j+1<<" ";
            cin>>data[i][j];
        }
    }
    cout<<"Total sales amount of each item\n";
    float sum;
    for(int i=0; i<cols; i++){
        sum = 0;
        for(int j=0; j<rows; j++) sum +=data[j][i];
        cout<<"Sales for item "<<i+1<<" = "<<sum<<endl;
    }
    for(int i=0; i<rows; i++)
        delete [] data[i];
    delete [] data;
    return 0;
}
```

Unit Five

C++ Functions

Summary

In order to solve complex problem by computer to achieve a desired objective, there are smaller problems needs to be solved before we solve the bigger problem. This the common approach while solving any real world problem. For example, to start some business, getting license, renting building for the business, setting up the environment, purchasing basic assets, employing staffs, cost estimation for service are considered as sub problems. Similarly, computer based solution implementation require identification of smaller problems and solving them independently and use them to solve the main problem. The smaller problems can be independently solved and their result can be communicated to other smaller problems or more complex problems. This unit tries to provide students a direction to address such issues to solve more complex problems. This is called modularization. Modularization facilitate team works, debugging solution from logical or run time error, maintenance of software, etc

General Objective

The purpose of this unit is to introduce students the concepts of modularization, which is vital to solve complex problems.

Specific Objectives

At the end of this unit, students should be able to:

- Explain what modularization is
- Explain stepwise bottom-up modularization and stepwise top-down modularization
- Define what a function is
- Explain how function is declared and defined
- Explain why function in C++ needs to be declared
- Distinguish the difference between declaration and definition of functions
- Describe the concept of reusing code/modules/function
- Identify and explain signature of a function
- Explain about arguments of a function
- Explain about parameter passing techniques in function implementation

- Identify parameter passing by value, reference, by address
- Explain how array/pointer can be passed as a parameter (one or more dimension)
- Distinguish the difference between formal and actual arguments
- Distinguish between functions whose return type is **void**, (**void ***) and other type
- Distinguish between functions whose argument is (**void ***) or (**type ***) for some other type
- Explain how to call a function from another function
- Distinguish the difference between recursive and non-recursive function call
- Distinguish the difference between global and local variables
- Identify scope of variables declared inside a function
- Distinguish the difference between automatic and static variables
- Explain what an inline function is
- Explain how to implement a function that support default value to its one or more arguments
- Identify what an overloaded function is
- Construct an overloaded functions

Self-test Questions

1. Write a C++ function that swaps the value of arguments so that the arguments value will be swapped upon completion of the function execution.
2. Write a C++ function that takes array of integer and its size from a calling function and displays the sum and average within the function body and doesn't return any value back to the caller.
3. Write a C++ function that takes a vector of real numbers as an argument and returns the norm of the vector using the function name.
4. Write a C++ function that takes a positive integer n as an argument and returns the largest power of two greater than or equal to n.
5. Write a C++ function that takes a positive integer n as an argument and returns 1 if n is prime and 0 otherwise.

6. Write a C++ function that takes a positive integer n as an argument and returns the next prime number greater than n
7. Write a program that compute the N^{th} prime number and display the result back to the user. (Use the above two functions)
8. Write a C++ function that takes a positive integer as input and returns the sum of the digits in its decimal representation. For example, the sum of the digits of 234567 is 27.
9. write a recursive function that takes n as an argument and return the sum of the first n integers
10. Write a C++ function that takes a positive integer as input and returns the equivalent number written from right to left. For example, the equivalent number for 234567 when it is written from right to left is 765432.
11. Write a function “no repetition(...)” which removes all repetition of characters from a string. Test the function in a suitable main program, which should be able to reproduce the following input/output:

Type in a string: **This string contains repeated characters**
The string without repetition is: **This trngcoaepd**
12. Write a C++ function that takes three integer variable (a , b , c) in as separate parameters and rotates the values stored so that value of a goes to b , value of b goes to c and c to a .
13. Write a modular program that solves the tower of Hanoi problem

Questions with answer key

1. Write a C++ function that takes 3 integer numbers as an input argument and returns the min and the max of the numbers as an argument.
2. Write a C++ function that takes a positive integer as input and returns the leading digit in its decimal representation. For example, the leading digit of 234567 is 2.
3. Write a C++ function that takes array of integers and its size and returns the sum and the average to the calling function through its parameters.
4. Given N which is the number of students who registered for 5 similar courses where each course has specific credit hours. Write a program that accept letter grades for each

student and display the all the students semester GPA on the screen. (Use pointers as required to optimally use memory)

Answer key

```
1.
void min_max(int val1, int val2, int val3, int *min, int *max){
    *min = (val1 < val2 && val1 < val3? val1 : (v2 < val3 ? val2 : val3));
    *max = (val1 > val2 && val1 > val3? val1 : (v2 > val3 ? val2 : val3));
}
```

```
2.
int leadingDigit(int value){
    int divisor = 1;
    float fValue = value;
    while (fValue / divisor >= 1) divisor *=10;
    return value/divisor;
}
```

```
3.
void statistics(int data[], int size, int *sum, float *average){
    int temp = 0;
    for(int i=0; i<size; i++) temp = temp+data[i];
    *sum = temp;
    *average = (float(temp)/size);
}
```

```
4.
#include<iostream.h>
void getGrade(int N, grade);
void displaySGPA(float * SGPA, int N){
    cout<<"Semester GPA of all students "<<endl;
    cout<<"Student No\t\tSGPA"<<endl;
    for(int i=0; i<N; i++){
        cout<<i+1<<"\t\t"<<SGPA[i]<<endl;
    }
}
float * getSGPA(int N, int ** grade,int total_credit){
    float * sgpa = new float[N];
    for(int i=0; i < N; i++){
        sgpa[i] = (grade[0][i] + grade[1][i] + grade[2][i] + grade[3][i] + grade[4][i]);
        sgpa[i] /=total_credit;
    }
    return sgpa;
}
int main(){
    int * grades[5]; //grades of the 5 courses for each students
    float *SGPA;
    int N, credit[5] = {3,3,3,3,3}; //assume all courses have three credit but can be changed
    cout<<"How many students you have -->"; cin>>N;
    getGrade(N, grade);
    int total_credit = credit[0]+ credit[1]+ credit[2]+ credit[3]+ credit[4];
    SGPA = getSGPA(N, grade, total_credit);
    displaySGPA(SGPA, N);
}
void getGrade(int N, grade){
    for(int i=0; i <5; i++){
        grades[i] = new int[N];
        cout<<"Enter grades in capital letter of all the students for" cout<<i+1<<endl;
        for(int j=0; j< N; j++){
            do{
                int valid = 1;
                cout<<"Grade of student "<<j+1<<" = [A or B or C or D or F] "; cin>>ch;
                switch(ch){
                    case 'A': grade[i][j] = A; break;
                    case 'B': grade[i][j] = B; break;
                    case 'C': grade[i][j] = C; break;
                    case 'D': grade[i][j] = D; break;
                    case 'F': grade[i][j] = F; break;
                    default: cout<<"Invalid grade entered"; valid = 0;
                }
                while(valid==0);
            }
        }
    }
}
```

Unit Six

File Processing in C++ Programming

Summary

While solving problem, input data to the program should be provided to the program and placed in the structured designed to hold data. These data is usually large in size cannot be entered into the system from the keyboard as we did in the previous units. Real problem data may even be used again and again by the same or different application programs. Hence, file input and output is the most appropriate mechanism for most computer based application to be usable. This unit is intended to provide guideline to students so that they will address file processing issue.

General Objective

The purpose of this unit is to introduce student file processing and file management concepts which are vital to solve real world application that demand large input from the environment and output its processing result permanently for future use.

Specific Objectives

At the end of this unit, students should be able to:

- Explain how file can be used as an input device
- Explain how file can be used as an output device
- Explain what stream mean in C++ input/output process
- Distinguish istream and ostream objects in C++
- Distinguish ifstream and ofstream objects in C++
- Identify the limitations of getting input from the keyboard
- Identify the limitations of sending output to the screen
- Explain how to declare input/output file handle for use in C++ program
- Identify the different modes of opening a file
- Distinguish the difference between binary and text file mode
- Explain how to open a file for the specified mode and assign file handle to the appropriate file handle variable

- Use both text and binary file to read its content
- Use both text and binary file to update its content
- Use both text and binary file write into it
- Use the functions that can be used to determine the state of the file opened
- Explain how to seek into a specific location of a file stream
- Explain why one should close a file

Self-test Questions

1. Write a program that accept from a user a line of text and write it back to a file called test.txt under your preferred directory. If the file exist, don't erase the previous content rather append the new text at the end
2. Write a program that reads the content of the file test.txt created in question number 1 and display into the screen
3. Write a program that accept N floating point number from the user and write first N and the N numbers into a file called numers.txt under your preferred directory. If the file exist, respond for the user and stop reading and writing the numbers
4. Repeat the above question so that the file is written as a binary file with filename numbers_in_binary.txt
5. Which one demand larger file size (numers.txt or numbers_in_binary.txt)
6. Write a program that reads the file you created in Question number 3 and display the data value into the screen
7. Write a program that reads the file you created in Question number 4 and display the data value into the screen
8. Write a program that accept a text filename from the user through the keyboard and count the frequency of all the characters and display all the distinct characters with frequency greater than zero together with their frequencies.
9. Write a program that takes two file names as argument and returns true (1) if the content of the two files are identically the same, else returns false (0) and write a main function that implements this function
10. Write a program that first takes two file names filename1 and filename2. The program them copies the content of the first file into the second.

11. Write a program that writes into a file `matrix.txt` the Matrix A and B each contains NxM dimension of matrix of data where each elements are integer. The program should write first N (number of rows) followed by M (number of columns) then the data content of both A and B row by row.
12. Write a program that reads the file content created at question number 11 and display the sum of the two matrices into a new file call `sum.txt` with the same format as `matrix.txt`

Questions with answer key

1. Write a program that define array of N character as a buffer ($N > 1000$) and that accept a line of text whose size is smaller than 100. The line of the string will be added into the buffer as far as the buffer has space to hold the entire characters of the string. Otherwise the buffer should be cleared by writing its content into a file called `buffer.txt` and clear the buffer for the intended purpose. The user input terminates while he input a zero length string.

Answer key

```
1.
#include<iostream.h>
#include<fstream.h>
#include<string.h>
#define MAX 1000
int main(){
    ofstream outf;
    char buffer[MAX], str[100];
    int bufferlen = 0;
    outf.open("d:\\buffer.txt", ios::out);
    do{
        cout<<"Enter your string "; cin.getline(str, 100, '\n');
        len = strlen(str);
        if(bufferlen + len >= MAX){ //clear the buffer
            outf<<buffer;
            bufferlen = 0;
            strcpy(buffer, str);
        }
        else{
            strcat(buffer, str);
            bufferlen +=len;
        }
    }
    outf<<buffer<<endl;
    outf.close();
    return 0;
}
```


Unit Seven C++ Structure and Class basics

Summary

We have seen that array and pointers gives as a mechanism to structure input data for use in complex computational procedure. These structure permits us to structure data if the elements come from the same data types. However, real world data are highly complex and elements may not have the same type. Moreover, the previous discussion shows both data and operations are interleaved together which expose data to be managed by any part of your program code. Hence, we may need to separate data and its operation to avoid unauthorized access to the data. This unit is intended to provide guideline to students so that they will address how such complex data can be structured in more appropriate and natural fashion.

General Objective

The purpose of this unit is to introduce student about structures and class basics in C++ which are vital for structuring complex format of data elements consists of various types of basic data elements and with appropriate data protection from being accessed through unauthorized module of the program.

Specific Objectives

At the end of this unit, students should be able to:

- Identify enumerated type in C++
- Use enumerated type in C++ program
- Explain how to define structure in C++
- Explain how a C++ structure memory is arranged
- Explain how to define variable identifier from a structure defined
- Explain how to use structure in declaration statement, parameter passing, dynamic memory allocation and where ever we need data type to be specified
- Identify how to identify an element from a structure variable
- Identify the concept of class in C++
- Explain how to define class in C++
- Explain how a C++ class memory is arranged

- Explain how to define variable identifier from a class (called object)
- Distinguish structure and class in C++ program
- Distinguish Class and Object in C++
- Distinguish the different access modifiers in Object Oriented Programming
- Explain what a constructor and destructor mean in C++ class definition
- Explain how to use class in declaration statement and parameter passing

Self-test Questions

1. What is a structure? Describe the role of a structure to model real world data for programming
2. What is class? Describe the role of class to model real world object
3. What member variables in a structure?
4. What is member variable and member function in a class definition
5. What is data hiding in Object Oriented Programming
6. What is member function in Object Oriented Programming
7. What are private and public access modifiers in Object Oriented Programming
8. Define a structure called Point that define a point in 3D space
9. Define a structure called Vector that define a vector of integer values and its dimension
10. Write a program that accept dimension of two vectors and their elements value from the user so that the program will generate the sum of the two vectors (use the structure defined above)

Questions with answer key

1. Create a class called **VectorClass** that define a vector of floating point with the specified dimension that supports the following operation:
 - Vector Addition
 - Vector difference
 - Dot product of vectorsThe class should have:
 - A constructor that accept dimension and set all its elements value 0.

- A constructor that accept dimension and an array of floating point values and initialize its elements with the array values
 - A destructor that release the memory space dynamically allocated for the vector
2. Write a program that implements the VectorClass to Read two points from the keyboard and display their sum, Difference and Dot Product.

Answer key

1.

Class definition

```
#include<iostream.h>
class vector{
    private:
        float *Vector;
        int dimension;
    public:
        vector(int n);
        vector(int n, float *data);
        ~vector();
        int getDimension();
        float *getVector();
        vector vector::addition(vector *v);
        vector vector::difference(vector *v);
        vector vector::multiplication(vector *v);
        float vector::dotproduct(vector *v);
        void display();
};
```

Constructors and destructor

```
vector::vector(int n){
    dimension = n;
    Vector = new float[dimension];
    for(int i=0; i < dimension; i++) Vector[i] = 0.0;
}
vector::vector(int n, float *data){
    dimension = n;
    Vector = new float[dimension];
    for(int i=0; i < dimension; i++) Vector[i] = data[i];
}
vector::~~vector(){
    delete [] Vector;
}
```

Accessor

```
int vector::getDimension(){
    return dimension;
}
float * vector::getVector(){
    float *data = new float[dimension];
    for(int i = 0; i < dimension; i++) data[i] = Vector[i];
    return data;
}
```

Other member functions

```
void vector::display(){
    for(int i = 0; i < dimension; i++)
        cout<<Vector[i]<<" ";
    cout<<endl;
}
vector vector::addition(vector *v){
    float *data = v->getVector();
    for(int i=0; i < dimension; i++)
        data[i] = data[i] + Vector[i];
    vector *result = new vector(dimension, data);
    delete[] data;
    return (*result);
}
vector vector::difference(vector *v){
    float *data = v->getVector();
    for(int i=0; i < dimension; i++) data[i] =Vector[i] - data[i];
    vector *result = new vector(dimension, data);
    delete[] data;
    return (*result);
}
vector vector::multiplication(vector *v){
    float *data = v->getVector();
    for(int i=0; i < dimension; i++) data[i] =Vector[i] * data[i];
    vector *result = new vector(dimension, data);
    delete[] data;
    return (*result);
}
float vector::dotproduct(vector *v){
    float sum = 0.0;
    float *data = v->getVector();
    for(int i=0; i < dimension; i++) sum += Vector[i] * data[i];
    delete[] data;
    return sum;
}
```

```
2
int main(){
    int dim;
    cout<<"Please enter the dimension of the two vectors "; cin>>dim;
    float * data = new float[dim];
    cout<<"Please enter the first vector "<<endl;
    for(int i = 0; i < dim; i++)
        cin>>data[i];

    vector *v1 = new vector(dim, data);
    v1->display();
    cout<<"please enter the second vector "<<endl;
    for(int i = 0; i < dim; i++)
        cin>>data[i];

    vector *v2 = new vector(dim, data);
    v2->display();

    vector v3 = v1->addition(v2);
    cout<<"The sum of the vector is \n\t\t";
    v3.display();
    vector v4 = v1->difference(v2);
    cout<<"The difference of the vector is \n\t\t";
    v4.display();
    vector v5 = v1->multiplication(v2);
    cout<<"The product of the vector is \n\t\t";
    v5.display();
    float v6 = v1->dotproduct(v2);
    cout<<"The dot product of the vector is \n\t\t"<<v6<<endl;
    return 0;
}
```

St. Mary's University College
Faculty of Informatics

Study Guide

For

Internet Programming

Department of Computer Science
Faculty of Informatics



Prepared by
Michael Melese

January 2012

Internet Programming

The client server communication established with the help of scripting (both client and server side) and markup languages following all the communication protocols and transmission protocols can be described as Internet programming.

Summary of the Module

This guide will get you up and running HTML, Cascade Style Sheet (CSS), JavaScript and Server Side Scripting Language (SSSL) specifically PHP so as to makes you ready for developing a dynamic website using MYSQL as a middleware.

Part I: Introduction to HTML and Client side Scripting Language

Unit I: HTML

HTML is a special kind of text document that is used by Web browsers to present text and graphics. HTML documents are often referred to as "Web pages". The browser retrieves Web pages from Web servers over the Internet without limitation.

Many people still write HTML by hand using tools such as Notepad on Windows, or gEdit on the Linux or using a third party Editor called WYSIWYG.

Once you are comfortable with the basics of HTML, you may want to learn how to add a touch of style using CSS, and to go on to try out features covered in my page on advanced HTML

General Objective

The general objective of this section is to make the student to learn about HTML so as to develop in order to develop a fully fledged static and dynamic site to be hosted on the internet.

Specific Objectives

The specific objective of this document is to learn HTML tags and basic document structure which includes Headings, Paragraphs, Formatting, Fonts, Styles, Links, Images, Tables, Lists, Forms, Color names and value and other HTML elements.

Self-test question

1. How can you open a link in a new browser window?
 - a. ``
 - b. ``
 - c. ``
 - d. ``
2. Which of the following path is supported by HTML?
 - a. Relative only
 - b. Absolute only
 - c. Absolute and Relative
3. There are _____ color names recognized by all version of HTML.
 - a. 6
 - b. 8
 - c. 256
 - d. 16
4. Interpret this statement: `SMUC`
 - a. It makes SMUC strong
 - b. It will print out SMUC in bold font
 - c. It highlights SMUC as being strong
5. What is the HTML? Discuss in detail
6. How can we add a comment in HTML? Show with an example.
7. What is Empty HTML element or tag? Give two examples of empty tag.
8. How do we define HTML Attribute Reference?
9. How Cell Padding is different from Cell Spacing?
10. How you display a picture in background of HTML?
11. How can we use Form's Action Attribute and Submit Button in HTML?
12. What are the different types of list and how they differ one from the other? List them and give one example for each.
13. Create the following table using HTML?

	Column 1	Column 2	Column 3
Row 1	row 1, cell 1	row 1, cell 2	row 1, cell 3
Row 2	row 2, cell 1	row 2, cell 2	row 2, cell 3

14. How can we achieve text formatting in HTML? Give at least 5 examples of text formatting tag and discuss how to use inside the code.
15. Suppose you need to display a text without any modification including the newline, space and the like element. What type of tag will allows you to construct this? Show with an example.
16. Construct the following table as it is using HTML.

St. Mary's University College
Faculty of Informatics

Main Table			
Sub Table One		Sub Table Two	
Item 1	Item 2	Item 3	Item 4

17. Given the code construct the table.

```
<html>
<head> <title>messed up table</title> </head>
<body>
<table width="400" border="1" align="center">
  <tr><td width="139" height="42"><del>Abysinia</del> </td>
    <td colspan="2" align="center">Ethiopia</td>
    <td colspan="2" rowspan="3">&lt;script language="&quot;javaScript&quot;&gt;<br />
      alert(&quot;can u see me?&quot;);<br />
    &lt;/script&gt;</td>
    <td width="80" rowspan="3" valign="top" align="right">TCP/IP Model </td>
    <td width="94">E<sub>X</sub>I<sup>T</sup>EXAM</sup> </td>
  </tr>
  <tr> <td colspan="2" align="center">The law of Inertia </td>
    <td width="58" rowspan="2">Pissa</td>
    <td rowspan="2"><pre> this text<br>
    will be diplayed<br> as it is using<br> <b> predefined tag</b> </td>
  </tr>
  <tr><td colspan="2" rowspan="2" align="right">How many colums and rows are
spanned plus aligned to the right </td>
  </tr>
  <tr> <td height="37" colspan="3" align="right">Lucy origin of life </td>
    <td colspan="2">R<sub>e</sub>n<sup>aissance</sup> </td>
  </tr>
  <tr><td rowspan="2">S&copy;hool of &copy;omputing</td>
    <td width="65">&nbsp;</td>
    <td colspan="3">Exchange rate for &pound;; </td>
    <td>SMUC</td>
    <td align="right">COSC</td>
  </tr>
  <tr><td colspan="2">&reg; trade mark </td>
    <td colspan="2" align="center"><b><i>bold, italic &amp; Justify</i></b> </td>
    <td colspan="2" align="right">HTML</td>
  </tr></table></body></html>
```

Solution for Self-test question

1. A
2. C
3. C
4. B
5. HTML stands for Hyper Text Markup Language. It is a markup language which is use to develop web pages. Using them we can describe our document into the form of text based information. For achieving those goals it has many tags like title, head, paragraph, table etc. In HTML we write tags surrounded with angle brackets. We can you some scripting language like JavaScript to display HTML doc in more attractive way.

Some main features of HTML are given below:

1. It is markup language.
2. It has provided pre-defined tags.
3. We save HTML file by using one of two extensions .htm or .html.
4. It uses markup tags to display on web pages.
6. You can add comment in HTML using `<!-- Your comment's goes here -->` as shown code below.

```
<html>
<body>
    <!--This comment will not be displayed-->
    <p>this is a regular paragraph</p>
</body>
</html>
```

7. When HTML element does not have any content to put between the starting and ending tag we will leave the tag empty. Then those elements are called as Empty element. Empty element does not have any end tag.
 `
` used to insert a new line
 `<hr>` used to insert a line
 `<input type="text" name="fname" >` used for displaying a text box for input element.
8. Here I have given you some example of attribute with their description. These are a simple tooltip to be displayed for the user as he/she moves the mouse over the link.

```
Syntax: :<a href="URL" title="sometitle"> .....</a>
<html>
<head>
    <title>Wel-Come to IP</title>
</head>
<body>
    <a    href="Internet    Programming.html"    title='IP'>Internet
Programming</a>
```

```
</body>
</html>
```

9. The difference between Cell-padding and Cell-spacing attribute of the table are discussed in detail below:

Cell Padding: Cell Padding in HTML: is used to define the how much space need between cell content and cell edges.

Syntax:

`<Table width="200" border="1" cellpadding="3">` means that it takes 3 pixels of padding inside the each cell.

Cell Spacing: It is also used to format the table .but it differ than cell padding because in cell padding we can set an extra space to separate the cell content with cell edges. Where as we use cell spacing to set the space between the cells.

Syntax:

`< Table width="200" border="1" cellspacing="10">`

We can use cell padding and cell spacing together like that,

`< Table width="200" border="1" cellpadding="3" cellspacing =”10”>`

10. We can display a picture as a background of HTML either using as a style of by using the background attribute of the page using as the code shown below.

Syntax:

`<body style="background-image:url(image path)">` or

`<body background="image path">`

11. Using Submit Button we can sends the form to the server and Action Attribute is specified that which file we send.

```
<form name="input" action="submit_form.php" method="get">
    <input type="hidden" name="hidden_val">
    <input type="text" name="Form">
    <input type="submit" value="Submit">
</form>
```

When we click the submit button web page than the sumbit_form is send to the server page of submit_form.php.

12. In HTML we can create ordered, unordered and definition list.

Ordered List: Ordered list define as list of items and they are fully numbered. Our ordered list starts with `` tag and end with `` tag.

```
<ol>
    <li>Internet Programming</li>
    <li>Rapid Application development</li>
    <li>Network Administration and Security </li>
</ol>
```

St. Mary's University College
Faculty of Informatics

Output:

1. Internet Programming
2. Rapid Application development
3. Network Administration and Security

Unordered List: Unordered list define as list of items and they are write with small black circles. Our ordered list starts with tag and end with tag.

```
<ul>
    <li>Internet Programming</li>
    <li>Rapid Application development</li>
    <li>Network Administration and Security </li>
</ul>
```

Output:

- Internet Programming
- Rapid Application development
- Network Administration and Security

Definition List: It is not only a list of items. In create list of item with their description. Description list start with <dl> tag and end with <dt> tag. In this we list each item by using <dt> and write their description by using <dd> tag as shown below.

```
<dl>
    <dt>Coffee</dt>
    <dd>Black Hot drink</dd>
    <dt>Ice Cream</dt>
    <dd> A smooth, sweet, cold food prepared from a frozen mixture of milk
    products.</dd>
</dl>
```

Output:

Coffee

Black Hot drink

Ice Cream

A smooth, sweet, cold food prepared from a frozen mixture of milk products.

You can use any kind of tag inside the above three type of list. i.e <p>,
, , <table>, <form>, headings and ...

13. We use <table> tag to create the table in HTML. After creating a table we use <tr> tag to create the rows and use <td> to create data cell on each row. Make sure to use <th> tag if the table contain heading. These cells can store images, text, tables, list etc.

Basic syntax to create an Table are given below:

```
<table border="1" align="Center">
```

St. Mary's University College
Faculty of Informatics

```
<tr>
    <th>&nbsp;</th>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
</tr>

<tr>
    <th>Row 1</th>
    <td>row 1, cell 1</td>
    <td>row 1, cell 2</td>
    <td>row 1, cell 3</td>
</tr>

<tr>
    <th>Row 2 </th>

    <td>row 2, cell 1</td>
    <td>row 2, cell 2</td>
    <td>row 2, cell 3</td>
</tr>
</table>
```

14. Using Text formatting we can change the overall view of the sentence like as shown below:

 This is bold text

<i> *This is a italic text*

<sub> Used to make the text subscripted like Co₂

<sup> Used to make the text superscripted like X²

<u> used to make the text underlined

<h1> Used to make the text as a header with largest font

Or you can use one or more combination (nested tag) of the above text in order to get different ways of formatted text.

15. <html>
 <body>
 <pre>
Text in a pre element tags are displayed
as you Typed
in a fixed-width
font, and it preserves

St. Mary's University College
Faculty of Informatics

both spaces and line breaks and Others

```
</pre>
```

<p>The pre element is often used to display computer code:</p>

```
<pre> for($i=0;$i<=100;$i++)  
        print($i);
```

```
</pre>
```

```
</body>
```

```
</html>
```

The output is shown below

```
Text in a pre element      tags   are   displayed  
as      you Typed  
in a fixed-width  
font, and it preserves  
both      spaces and line breaks      and Others
```

The pre element is often used to display computer code:

```
for ($i=0;$i<=100;$i++)  
    Print ($i);
```

16.

Code solution

```
<html>  
  <head>  
    <title> Table Construction </title>  
  </head>  
  <body>  
    <TABLE border=2 align="center">  
      <TR> <TD COLSPAN=3 ALIGN="center"><H2>Main Table</h2></TD>  
    </TR>  
    <TR>  
      <TD ALIGN="center">  
        <TABLE BORDER=4>  
          <TR> <TD COLSPAN=2 ALIGN="center">Sub Table One </TD>  
        </TR>  
          <TR> <TD>Item 1          </TD> <TD> Item 2 </TD> </TR>  
        </TABLE>  
      </TD>  
      <TD> </TD>  
      <TD ALIGN="center" >  
        <TABLE BORDER=4>  
          <TR> <TD COLSPAN=2 ALIGN="center">Sub Table Two </TD>  
        </TR>  
          <TR> <TD>Item 3          </TD> <TD> Item 4 </TR>  
        </TABLE>  
      </TD>  
    </TR>  
  </TABLE>  
  </body>
```

St. Mary's University College
Faculty of Informatics

</html>

17. Output the table code

Abyssinia	Ethiopia		TCP/IP Model	EXI ^T EXAM
The law of Inertia		<script language="javaScript"> alert("can u see me?"); </script>		this text will be diplayed as it is using predefined tag
How many colums and rows are spanned plus aligned to the right	Pissa			
		Lucy origin of life	Ren ^a aissance	
S©hool of ©omputing		Exchange rate for £;	SMUC	COSC
	® trade mark	<i>bold, italic & Justify</i>		HTML

Self-test question

- What type of protocol we are expected to use in order to access/send a file from/to a web server?
- Identify the correct order of HTML tags that used to construct a single page?
 - <html><head><title><title></head><body></body></html>
 - <html><head></head><body><title><title></body></html>
 - <html><head><title><title><body></body></head></html>
 - <html><title><title><head></head><body></body></html>
- Which command we use to link a page with another page of HTML page?
 -
 -
 -
 -
- Identify the correct tag that can be used to add an Image to a given page?
 - <image src="url">
 -
 - <src img="url">
 -
- Identify one or more elements of World Wide Web (www) that must be present in order to access a page from a remote location like from Google server?
 - Browser
 - Internet connection

St. Mary's University College
Faculty of Informatics

- c. Web server in the client side
 - d. Domain name system
 - e. HTTP protocol
 - f. The page itself
6. Choose the correct HTML tag for the largest heading
- a. <h6>
 - b. <heading>
 - c. <head>
 - d. <h1>
7. How can you create an e-mail link?
- a. <mail href="someone@smuc.edu.et">
 - b. <mail>someone@smuc.edu.et</mail>
 - c.
 - d.
8. Write the output of the following code
- ```
<html>
<body>
<pre> the quick brown for
 Jumps o<i>ve</i>r the lazy dog</pre>
<p>sample java for loop statement</p>
<pre>

 for(int i=0;i<=100;i++)
 document.out.println(i);

</pre>
</body>
</html>
```
9. Construct the following form using HTML. Make sure that the room reserved for should be selected by making the room type and reserved for disable so that the user can not select “room type” and reserved for option.

**St. Mary's University College**  
**Faculty of Informatics**

Booking Information			
<b>Customer Detail</b>			
First Name	<input type="text"/>	Middle Name	<input type="text"/>
Family Name	<input type="text"/>	Company	<input type="text"/>
Country/State	<input type="text"/>	City/Town	<input type="text"/>
Post/Zip Code	<input type="text"/>	Telephone	<input type="text"/>
E-Mail	<input type="text"/>		
Room Type	Room Type ▾	Reserved for	Reserve for ▾
Number of days	<input type="text"/>	<div style="border: 1px solid black; padding: 2px;"> Reserve for Family Single </div>	
<b>Additional Information</b>			
<input type="button" value="Register"/>		<input type="button" value="Clear"/>	

10. Write the difference and similarity between Relative and Absolute addressing system and give one example for each. Which addressing system is appropriate? Why?
11. Write the HTML code that displays the following table.

PC		HW		V <sub>g</sub>	
Graduating Class		Informatics			
CSS & Javascript				SW	
Server	2003	Labratory	S <sup>erver</sup> configuration		
		ICT Unit			

12. List the different elements of www and discuss each of them in detail.
13. Design your own personal site using HTML.

## Part II: Cascade Style Sheet (CSS)

### General Objective

The general objective of this section is to make the student to learn about Cascade Style Sheet so as to separate the presentation from the actual content of the document and to give the same feel and look among the entire website to be hosted on the internet.

**St. Mary's University College**  
**Faculty of Informatics**

**Specific Objectives**

The specific objective of this document is to learn the basic and advanced Cascade Style Sheet which includes an Inline, Internal and External style sheet and discuss the importance of structure/style separation, order and cascading mechanism

## **Self-test question**

1. Where in an HTML document is the correct place to refer to an external style sheet?
  - a. In the <body> section
  - b. At the end of the document
  - c. At the top of the document
  - d. In the <head> section
  - e. Between head and body
2. Which HTML tag is used to define an internal style sheet?
  - a. css
  - b. text/style
  - c. style
  - d. script
3. Which is the correct CSS syntax?
  - a. body {color: black}
  - b. body:color=black
  - c. {body:color=black(body}
  - d. {body;color:black}
4. How do you insert a comment in a CSS file?
  - a. /\* this is a comment \*/
  - b. ' this is a comment
  - c. // this is a comment //
  - d. // this is a comment
5. Which property is used to change the background color in CSS?
  - a. bgcolor:
  - b. background-color:
  - c. color:
6. How do you add a background color for all "<h1>" elements?
  - a. all.h1 {background-color:#FFFFFF}
  - b. h1.all {background-color:#FFFFFF}
  - c. h1 {background-color:#FFFFFF}
7. Which CSS property controls the text size?
  - a. font-style
  - b. text-style
  - c. font-size
  - d. text-size
8. How do you make a list that lists its items with squares?
  - a. type: square
  - b. list-style-type: square - correct answer
  - c. style-list: square

**St. Mary's University College**  
**Faculty of Informatics**

- d. list-type: square
9. Suppose you want to insert an image using CSS to the page, select the one that is appropriate to implement using an inline style sheet.
- background-image=url("picturepath")
  - background-image:url("picturepath")
  - background:url("picturepath")
  - Background=url("picturepath")
  - image:url("picturepath")
10. Write down at least three benefit that we can achieve from CSS.
11. What is the difference between CSS and HTML?
12. List the three different types of CSS and discuss them in detail.
13. Write a CSS code that change the background color of page to yellow, cursor type to crosshair and the text color to blue using inline, internal and External style sheet.
14. List the different type of selector in CSS and discuss them in detail.
15. What make one type of selector different from other types of selector?
16. What is the importance of having group selector?

**Solution for Self-test question**

1. D      2. C      3. A      4. A      5. B      6. C      7. C  
8. D      9. B

9. CSS allow us
- Control layout of many documents from one single style sheet;
  - more precise control of layout;
  - Apply different layout to different media-types.
  - Separates the presentation from the actual content of the data.
10. CSS and HTML are different due to the following
- HTML is used to structure content of document and CSS is used for formatting the structured content.
  - CSS is a style language that defines layout of HTML documents. covering fonts, colours, margins, lines, height, width, background images, advanced positions and many other things.
  - HTML can be mis-used to add layout to websites. But CSS offers more options and is more accurate and sophisticated

## 11. Different types of CSS

### In-line (the attribute style)

One way to apply CSS to HTML is by using the HTML attribute style.

```
<html>
<head>
 <title>Example</title>
</head>
<body style="background-color: #FF0000;">
 <p>This is a red page</p>
</body>
</html>
```

### Internal (the tag style)

Another way is to include the CSS codes using the HTML <style> tag between head tag.

```
<html>
<head>
 <title>Example</title>
 <style type="text/css">
 body {background-color: #FF0000;}
 </style>
</head>
<body>
 <p>This is a red page</p>
</body>
</html>
```

### External (link to a style sheet)

The recommended method is to link to a so-called external style sheet. An external style sheet is simply a text file with the extension **.css**. Like any other file, you can place the style sheet on your web server or hard disk.

For example, let's say that your style sheet is named **style.css** and is located in a folder named **style**.

Such link can be created with one line of HTML code:

```
<link rel="stylesheet" type="text/css" href="style/style.css" />
```

**St. Mary's University College**  
**Faculty of Informatics**

The line of code must be inserted in the header section of the HTML code i.e. between the <head> and </head> tags. Like this:

```
<html>
 <head>
 <title>My document</title>
 <link rel="stylesheet" type="text/css" href="style/style.css" />
 </head>
 <body>
 You html tag goes here
 </body>
</html>
```

This link tells the browser that it should use the layout from the CSS file when displaying the HTML file. Several HTML documents can be linked to the same style sheet. In other words, one CSS file can be used to control the layout of many HTML documents.

## 12. CSS implementation

### **Internal**

```
<html>
<head>
 <style type="text/css">
 body {
 background-color:#FFFF00;
 color:#0000FF;
 cursor:pointer;
 }
 </style>
</head>
<body>
 <h1>Mouse over me to see the cursor changing to pointer type</h1>
 <p>internal style sheet</p>
</body>
</html>
```

### **Inline**

```
<html>
 <body style="background-color:#FFFF00;
 color:#0000FF;cursor:pointer;">
```

**St. Mary's University College**  
**Faculty of Informatics**

```
<h1>Mouse over me to see the cursor changing to pointer type</h1>
<p>internal style sheet</p>
</body>
</html>
```

**External**

```
<html>
<head>
 <link rel="stylesheet" type="text/css" href=" style.css" />
</head>
<body>
 You html tag goes here
</body>
</html>

//Style.css
body {
 background-color:#FFFF00;
 color:#0000FF;
 cursor:pointer;
}
```

13. Type of selector

There are three types of selectors:

- **Tag selectors :**  
Used to define styles associated to specific HTML tags.
- **Class selectors**  
Used to define styles that can be used without redefining plain HTML tags.
- **ID selectors**  
Used to define styles relating to objects with a unique ID

14. The difference among the selector

- The difference between ID and class selector is that an ID selector can be called only once in a document, while a class selector can be called multiple times in a document.
- The second difference is that ID can be called by Javascript's getElementById function.
- Class Selectors are used to specify the same style for different tag and different style for the same tag.



**St. Mary's University College**  
**Faculty of Informatics**

- Class and ID names are both case sensitive. For example, .classone and .ClassOne are two different classes but not in tag selector.
- An ID selector applies styles to an element in the same way as a class.

15. The following is the importance of grouping

Helps to avoid repetition when applying the same styles to several html elements, you can group selectors as a comma-separated list. example

```
h1, h2, h3, h4, h5, h6 {
 color: #468966;
 font-family: Georgia, "Times New Roman", Times, serif;
 margin: 10px 0;
}
```

## **Self Assessment question**

1. What is the correct CSS syntax for making all the <p> elements bold?
  - a. p {text-size:bold}
  - b. p {font-weight:bold}
  - c. style:bold
  - d. p{font:bold}
2. How do you display a border like this: The top border = 10 pixels, The bottom border = 5 pixels, The left border = 20 pixels, The right border = 1pixel?
  - a. border-width:10px 20px 5px 1px
  - b. border-width:10px 1px 5px 20px
  - c. border-width:10px 5px 20px 1px
  - d. border-width:5px 20px 10px 1px
3. How do you change the left margin of an element?
  - a. padding:
  - b. indent:
  - c. margin:
  - d. text-indent:
  - e. margin-left:
4. What is the correct HTML for referring to an external style sheet?
  - a. <link rel="stylesheet" type="text/css" href="mainstyle.css">

**St. Mary's University College**  
**Faculty of Informatics**

- b. `<style src="mainstyle.css">`
  - c. `<stylesheet>mainstyle.css</stylesheet>`
  - d. `<link url="stylesheet" type="text/css" href="mainstyle.css">`
5. What is the correct CSS syntax for making all the `<p>` elements bold?
- a. `p {font-weight:bold}`
  - b. `p {text-size:bold}`
  - c. `<p style="text-size:bold">`
  - d. `<p style="font-size:bold">`
6. How do you make each word in a text start with a capital letter?
- a. `text-transform:uppercase`
  - b. `text-transform:capitalize`
  - c. `text-transform:togglecase`
7. What makes CSS different from HTML? Discuss in detail.
8. Arrange the CSS style from highest to the lowest precedence and which one is the most common style that we can apply for a page.
9. Given the following CSS group them by selecting the best match among the given attribute.

```
.right {
margin: 10px 1.2% 15px 1.2%;
padding: 5px 0.5% 5px 0.5%;
}
```

```
.left {
margin: 10px 1.2% 10px 1.2%;
padding: 5px 0.5% 7px 0.5%;
}
```

```
.box_header {
margin: 10px 1.2% 15px 1.2%;
padding: 5px 0.5% 7px 0.5%;
}
```

**St. Mary's University College**  
**Faculty of Informatics**

10. Is there a difference between the CSS shown below? Discuss the similarity and differences between the CSS style a and b.

a. `p, b, i {  
margin: 10px 1.2% 10px 1.2%;  
padding: 5px 0.5% 7px 0.5%;  
}`

b. `p b i {  
margin: 10px 1.2% 10px 1.2%;  
padding: 5px 0.5% 7px 0.5%;  
}`

11. Given the following attribute of style write a CSS code that will allow you to implement the CSS style of inline, internal and external style sheet and using the three types of selector along with the HTML tag.

- Tag selector
- Class selector
- ID Selector

Float: left, color: #000, height: 275px, font-size: 10px, background-repeat: no-repeat, background-position: left width: 100% and font-family: Verdana, Arial, sans-serif;

12. Given the following specification write a CSS code that will allow you to implement the CSS style for scroll-bar of a page

3dlight-color: red, arrow-color: yellow, base-color: blue, darkshadow-color: cyan,  
face-color: green, highlight-color: grey, shadow-color: pink and scrollbar-track-color: silver

13. Apply different CSS style for the personal page that you have designed in the first section of HTML.

## Unit I: Client side Scripting Language

JavaScript is scripting language used for client side scripting which is a dynamic typed language. JavaScript can be thought of as an extension to HTML which allows authors to incorporate functionality in their web pages by detect what browser a person is using, customize the pages to their browser, swap images on a page to create rollovers, perform calculations in forms and check to ensure that entries in the form are correct and prompt users for information and use it to customize a page.

### General Objective

The general objective of this section is to make the student to learn about JavaScript so as to develop an event-driven programming in response to events that occur in a Web browser.

### Specific Objectives

The specific objective of this document is to make use of JavaScript accomplished its tasks by using objects (things like windows, forms, string documents) methods and properties.

### Unit self-test question list with answer key

1. Identify the JavaScript code that display a random number between 5 and 9 inclusive is:
  - a) `Math.floor((Math.random() * 5) + 4);`
  - b) `Math.floor((Math.random() * 4) + 4);`
  - c) `Math.floor((Math.random() * 4) + 5);`
  - d) `Math.floor((Math.random() * 5) + 5);`
2. Which of the following JavaScript statements will assign the word "and" to variable s?  
`var str="Mix and Match"`
  - a) `s = str.substring(4,3);`
  - b) `s = str.substring(5,3);`
  - c) `s = str.substring(4,7);`
  - d) `s = str.substring(5,7);`
  - e) `s = str.substring(5,8);`
3. Write a JavaScript code that validate E-Mail Validation code given the following form.



Enter an Email Address :

4. Write a javascript code that validate Number, Text and Alphanumeric from the given input below

**St. Mary's University College**  
**Faculty of Informatics**

Enter a Number :	<input type="text"/>
Enter a text :	<input type="text"/>
Enter a text and Number :	<input type="text"/>
<input type="button" value="Submit"/>	

5. Write a java script program that changes the background color of the pages when the user selects the color from combo box.
6. List the different advantages of JavaScript, what it can do and what it can't do.
7. Write a Javascript program/function that shows the numbers [1-10] in an alert box, one after another sleeping for 100, 200, ..., 900 msecs. That is, when the program starts, it should show "1". Then it should sleep for 100 msec. and show "2", sleep for 200 msec. and show "3" and so on, finally showing "10" before returning.
8. write a JavaScript program that insert an image in the page with the <img> tag. When the mouse pointer rolls over the image, it should switch to a different image. When it rolls out (leaves the image), it should swap back again. You'll need to wrap the images inside <a> tags because img objects don't have the onmouseover and onmouseout events.

### Solution for self Assessment

1. D
2. C
3. Code for E-mail validation

```
<html>
<head>
<title>Email Validation</title>
<script language = "Javascript">
function Check_Email(Email){
 var LengthOfAt=Email.indexOf("@")
 var EmailLength=Email.length
 var LengthOfDot=Email.indexOf(".")
 if (Email.indexOf("@")==-1){
 alert("Invalid E-mail ID")
 return false
 }
 if (Email.indexOf("@")==-1 || Email.indexOf("@")==0 || Email.indexOf("@")==EmailLength){
 alert("Invalid E-mail ID")
 return false
 }
}
```

**St. Mary's University College**  
**Faculty of Informatics**

```
if (Email.indexOf(".")==-1 || Email.indexOf(".")==0 || Email.indexOf(".")==EmailLength){
 alert("Invalid E-mail ID")
 return false
}
if (Email.indexOf("@",(LengthOfAt+1))!=-1) {
 alert("Invalid E-mail ID")
 return false
}
if (Email.substring(LengthOfAt-1,LengthOfAt)=="." ||
Email.substring(LengthOfAt+1,LengthOfAt+2)==".") {
 alert("Invalid E-mail ID")
 return false
}
if (Email.indexOf(".",(LengthOfAt+2))==-1) {
 alert("Invalid E-mail ID")
 return false
}
if (Email.indexOf(" ")!=-1) {
 alert("Invalid E-mail ID")
 return false
}
return true
}
function ValidateForm(){
 var emailID=document.frmSample.txtEmail
 if ((emailID.value==null)||(emailID.value=="")) {
 alert("Please Enter your Email ID")
 emailID.focus()
 return false
 }
 if (Check_Email(emailID.value)==false){
 emailID.value=""
 emailID.focus()
 return false
 }
 return true
}
</script>
</head>
<body>
<form name="frmSample" method="post" action="#" onSubmit="return ValidateForm()">
 <table bgcolor="#FF0000" border="2" align="center">
 <tr>
 <td>Enter an Email Address : </td>
 <td><input type="text" name="txtEmail"></td>
 </tr>
 <tr><td colspan="2" align="center"><input type="submit" name="Submit"
value="Submit"></td>
 </tr>
 </table>
```

**St. Mary's University College**  
**Faculty of Informatics**

```
</form>
</body>
</html>
```

4. Codes for validation of Number, Text and Alphanumeric

```
<html>
<head>
<title>Number Validation</title>
<script language = "Javascript">
```

/\*It's also possible to create more complex validation routines using JavaScript's built-in support for regular expressions. Consider the following methods, which uses a regular expression to test whether all the characters or combination of the character and integer so as to validate the input element of the form... \*/

```
function isInteger(){
 s=document.numeric_validation.num.value;
 for (i = 0; i < s.length; i++){
 var c = s.charAt(i);
 if (((c < "0") || (c > "9"))) {
 alert("Not a Number");
 document.validation.num.value="";
 document.validation.num.focus();
 return false;
 }
 }
 return true;
}

function isAlphabetic(){
 val=document.validation.num.value;
 if (val.match(/^[a-zA-Z]+$/)){
 alert("Character");
 }
 else
 alert("Not A valid Text ");
}

function isAlphaNumeric(){
 val=document.validation.combine.value;
 if (val.match(/^[a-zA-Z0-9]+$/)){
 alert("Valid AlphNumeric Character");
 }
 else
 alert("Not A AlphNumeric Character");
}

</script>
</head>
<body>
```

**St. Mary's University College**  
**Faculty of Informatics**

```
<form name="validation" method="post" action="" >
<table width="280" border="1" align="center" bgcolor="#CC0000">
 <tr><td>Enter a Number :</td>
 <td><input type="text" name="num" size="15" onBlur="return isInteger(this)"></td> </tr>
 <tr><td>Enter a text :</td>
 <td><input type="text" name="text" size="15" onBlur="return isAlphabetic(this)"></td>
 </tr>
 <tr><td>Enter a text and Number :</td>
 <td><input type="text" name="combine" size="15" onBlur="return isAlphaNumeric(this)"></td>
 </tr>
 <tr><td colspan="2" align="center"> <input type="submit" name="Submit" value="Submit"></td>
 </tr>
</table>
</form>
</body>
</html>
```

5. codes for changing background of the page.

```
<form>
 <select onChange="document.bgColor=this.options[this.selectedIndex].value">
 <option value="40E0D0"> Torquoise
 <option value="2E8B57"> Sea Green
 <option value="87CEEB"> Sky Blue
 <option value="F4A460"> Sandy Brown
 <option value="FFF0F5"> Lavender Blush
 <option value="FFFFFF" SELECTED> White
 </select>
</form>
```

6. Solution for advantage and what a JavaScript can do

**Make your web pages responsive.**

Web environments are dynamic, things happen all the time: the web page loads in the browser, the user clicks a button or moves the mouse over a link, etc. These are called events. With JavaScript you can make the page immediately react to these events the way you choose: for example, by showing or hiding specific page elements, by changing the background color, etc.

**Detect visitors' browsers.**

You can use a JavaScript script to detect the visitor's browser, or, even better.

**Create cookies.**

A JavaScript script is great if you want to create cookies so that your visitors can enjoy a personalized experience the next time they visit your website.

**Validate web form data.**

You can use a JavaScript script to validate form data before the form is submitted to a server. This saves the server from extra processing.

7. Solution for time count-down

```
var a = 1;
```



**St. Mary's University College**  
**Faculty of Informatics**

```
function showNumbers(){
 alert (a++);
 if (a < 11)
 setTimeout ("showNumbers()", (a-1) * 100);
}
```

8. Solution for mouseOver and mouseOut event

```
<html>
<head>
<script language=JavaScript>
 function mouseOver(){
 document.images["myImage"].src = "Img2.jpg";
 }
 function mouseOut(){
 document.images["myImage"].src = "Img1.jpg";
 }
</script>
</head>
<body>
<a style="cursor : default" href="" onclick="return false" name="link1"
onmouseover="mouseOver()" onmouseout="mouseOut()">

</body>
</html>
```

### Unit self-test question list without answer key

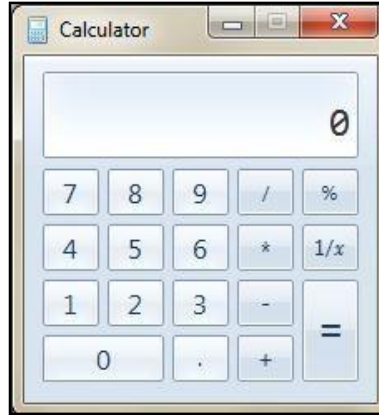
1. Write a java script program that displays in a table the number, square and cubic for a number between 1 and 150 using the following format.

Number	Square	Cube
1	1	1
2	4	8
3	9	27
.	.	.
.	.	.
150	22500	506250000

2. Write a JavaScript program that accepts a positive number from the user and calculate the product of the integer between 1 and the number that the user entered.
3. Write a JavaScript program that accept the result of the student out of 100 from the user and computes a letter grade based on the following rules: Using both if ... else and switch statement.

**St. Mary's University College**  
**Faculty of Informatics**

4. Mention the three different ways to implement JavaScript and discuss each of them.
5. Write a JavaScript program that can serve as a calculator having the following basic functionality as shown figure below.



6. write a JavaScript program that find the number of occurrences of each letter in specified string “addis ababa is the capital city of ethiopia”.
7. Write a JavaScript program that calculate the factorial of the number using a function.

Write the output of this code

```
<script language=javascript type="text/javascript">
 var x = 5;
 document.write("<p>x before function call = " + x);
 test();
 document.write("<p>x after function call = " + x);

 function test()
 {
 var x = 1000;
 document.write("<p>x in function = " + x)
 }
</script>
```

8. Write a JavaScript function that takes array of numbers as an argument. It should apply the function to each entry in the array, add up the results, and return the sum. So if you call yourNewFunction([1, 2, 3]), it should return 14 ( sum of the squares of 1, 2, and 3)
9. Write a JavaScript program that calculate the Fibonacci series.

**St. Mary's University College**  
**Faculty of Informatics**

Note : The **Fibonacci numbers** are the integer sequence 0, 1, 1, 2, 3, 5, 8, 13, 21, ..., **n** which each item is formed by adding the the previous two number. (**Hint Use recursive function**)

10. Write a JavaScript program that accept two numbers from the user and multiplies them and finally prints out the result on the screen.
11. Write a JavaScript program that read an integer value from the keyboard and displays a message indicating the number is odd or even.
12. Write a small JavaScript program that prints out the even integers from 0 to 100 using a while loop.
13. Using conditional statement and iteration control structure to output all the odd integers from 1 to 101 that are not divisible by either 3 or 5.(**use continue and break**)
14. Your program should output integers such as 1, 7, 11, 13, 19 ...
15. Write a JavaScript program that accept an integer value from the user and display the factorial of that number.  
$$\text{factorial}(n) = 1 \text{ if } n < 2, \quad n * (n - 1) * \dots * 1 \text{ otherwise}$$
16. Write an application that inputs an integer containing only 0s and 1s (i.e., a binary integer) and prints its decimal equivalent, the program must reject other inputs.
17. Write an application that displays a table of the binary, octal and hexadecimal equivalents of the decimal numbers in the range 1 through 256.
18. An integer is said to be prime if it is divisible by only 1 and the number itself. Write a program that accepts a positive integer and counts the number of primes which are less than the given number entered by the user. Display the count value of the primes less than the number.
19. Write JavaScript program that estimates the value of the mathematical constant e by using the formula:
  1.  $e = 1 + 1/1! + 1/2! + 1/3! + \dots$  compare its result with the Math constant Math.E
20. Write a JavaScript program that enables you to convert from degree Fahrenheit to Celsius and vise verse and finally displays the information on the screen using the following formula. (Hint use HTML form for accepting value and to what to convert)

$$\text{Fahrenheit} = 1.8 * \text{celcius} + 32$$

$$\text{Celcius} = (5/9) * \text{Fahrenheit} - 32$$

**St. Mary's University College**  
**Faculty of Informatics**

21. Write a program that defines two floating point variables. The first variable that represent the monthly income (salary) and the second variable for payable allowance of the employee. The question is to write a JavaScript program that accept the salary along with the allowance and finally calculate the amount of tax and net income based on the following condition

Sal <=15	tax= Sal *0%
Sal <=650	tax= Sal *10%-15
Sal <=1400	tax= Sal *0.15-47.5
Sal <=2350	tax= Sal *0.2-117.5
Sal <=3550	tax= Sal *25%-235
Sal <=5000	tax= Sal *30%-412.5
Sal >5000	tax= Sal *35%-662.5

- Total\_income = Sal + allowance, Net\_income = Sal +allowance - tax
- For the above question allow the user to enter the number of employee and finally display the average, maximum, and minimum statistics about employee salary, allowance and tax in formatted way for the user and make sure to include Escape Sequences for Special Characters a(backspace, tab, linefeed, Carriage return, double quote, Single quote and back space) using. (using while, do while and for loop)

## **Unit IV: PHP**

PHP (Hypertext Preprocessor) is one of open source server-side scripting language that executed on the server side supporting many databases like MySQL, Informix, Oracle, Sybase, Solid, etc.

PHP runs on different platforms Windows, Linux, UNIX, and are compatible with almost all servers used today and an easy to learn and runs efficiently on the server side.

## **General Objective**

The general objective of this section is to make the student to learn about PHP as one of the server side scripting language so as to develop a dynamic to be hosted on the internet under the Apache web server.

## **Specific Objectives**

The specific objective of this document is to learn the basic syntax, variable, conditional structure, array, string manipulation, function which might be user defined or built in, how to embed HTML and JavaScript and the CRUD operations using apache as a server and MYSQL as database management system.

**St. Mary's University College**  
**Faculty of Informatics**

**Unit self-test question list with answer key**

1. Is it possible to pass data from PHP to JavaScript?
  - a. No, because PHP is server-side and JavaScript is client-side.
  - b. No, because PHP is a loosely typed language.
  - c. Yes, because JavaScript executes before PHP.
  - d. Yes, because PHP can generate valid JavaScript.
2. Is it possible to pass data from JavaScript to PHP?
  - a. Yes, but not without sending another HTTP request.
  - b. Yes, because PHP executes before JavaScript.
  - c. No, because JavaScript is server-side, and PHP is client-side.
  - d. No, because JavaScript executes before PHP.
3. Which types of form elements can be excluded from the HTTP request?
  - a. text, radio, and check box
  - b. text, submit, and hidden
  - c. submit and hidden
  - d. radio and check box
4. When processing the form, what is the difference between a hidden form element and a non hidden one, such as a text box?
  - a. The hidden form element does not have a name.
  - b. There is no difference.
  - c. The hidden form element does not have a value.
  - d. The hidden form element is excluded from the request.
5. Which of the following form element names can be used to create an array in PHP?
  - a. myArray
  - b. [myArray]
  - c. myArray[]
  - d. myArray[bar]
6. Why must you call session\_start() prior to any output?

**St. Mary's University College**  
**Faculty of Informatics**

- a. Because it is easy to forget if not placed at the top of your scripts.
  - b. Because you can no longer access the session data store after there has been output.
  - c. Because session\_start() sets some HTTP headers.
  - d. Because calling session\_start() causes the HTTP headers to be sent.
7. Which of the following represents the proper way to set a session variable?
- a. `$_SESSION['foo'] = 'bar';`
  - b. `session_start();`
  - c. `session_set_save_handler ('myopen');`
  - d. `$foo = $_SESSION['foo'];`
8. The command responsible to create a link with the server for MySQL database operations
- a. `mysql_sql(username,password);`
  - b. `mysql_connect_db(server,username,password);`
  - c. `mysql_select_db(databaseName);`
  - d. `mysql_connect(server,username,password);`
9. Which one of the following is true about PHP ?
- a. PHP is one of the client-side scripting language, like JavaScript.
  - b. PHP is one of an open source software that supports only MYSQL databases.
  - c. PHP files are returned to the browser as plain HTML text.
  - d. can have a file extension of ".pph", ".pph3", or ".phmtl"
10. Select the correct statement(s) which is/are true forms in HTML and PHP
- a. Submit buttons always forwards the form element to the page specified in the action attribute of the form.
  - b. The METHOD attribute of the form specifies the names of form element
  - c. The values of form elements can be accessed in PHP pages by using their names
  - d. HTML and JavaScript element can be embedded in php script.
11. Which one of the following is true provided that we have selected appropriate MYSQL database from which we access all information
- `$res=mysql_query("select name from user") or die("failed");`
- a. \$res is a string that represent the query

**St. Mary's University College**  
**Faculty of Informatics**

- b. \$res is an array that contain record fetched from the table user
  - c. The command always die after successfully completing mysql\_query()
  - d. \$res holds a simple number of record
12. The command responsible to create a link with the server for MySQL database operations
- a. mysql\_select\_db(databaseName);
  - b. mysql\_connect(server,username,password);
  - c. mysql\_sql(username,password);
  - d. mysql\_connect\_db(server,username,password);
13. Which one of the following is **true** about validation?
- a. Validation is always performed at the client side.
  - b. Validation in the client side is preferable if the data is to be inserted in to a database.
  - c. Validation in the server side is preferable if the data is to not to be inserted in the server side.
  - d. Validation is always performed at the server side.
  - e. None of the above
14. Which one of the following is **false** about forms
- a. The METHOD attribute of the form specifies the names of form element
  - b. Submit buttons always forwards the form element to the page specified in the action attribute of the form.
  - c. The values of form elements can be accessed in PHP pages by using their names
  - d. HTML and JavaScript element can be embedded in php script.
15. What will be the output of the following script?

```
<?php
$array=array(1,2,3,5,8,13,21,34,55);
$sum=0;
for($i=0;$i<5;$i++)
 $sum+=$array[$array[$i]];
print($sum);
?>
```



**St. Mary's University College**  
**Faculty of Informatics**

- a. 5                      b. 0                      c. 19                      d. 78

Answer the following three questions based on the code given below along with each question

```
<?php
$con = mysql_connect("localhost","root","vertrigo");
if (!$con){
 die('Could not connect: ' . mysql_error());}
if (!mysql_query("CREATE DATABASE my_db",$con))
echo "Error creating database: " . mysql_error();
mysql_select_db("my_db", $con);
$sql = "CREATE TABLE Persons(FirstName varchar(15), LastName
varchar(15),Age int)";
$result=mysql_query($sql,$con);
mysql_close($con);
?>
```

16. Suppose you put // before the code `$result=mysql_query($sql,$con);` What would happen to the database after successful creation of the connection ?
- a. The code will be executed successfully and create a database but not the table.
  - b. An error is generated in creating database and table will be created.
  - c. The code will be executed successfully and create a table but not the database.
  - d. An error is generated without creating database as well as table.
17. Suppose the default username="vertrigo" and password="root" for MYSQL database. What would happen to the code if you provide the following code for creating a connection?
- ```
$con = mysql_connect("localhost", "root", "vertrigo");
```
- a. The code will be executed successfully and create a database but not the table.
 - b. An error is generated in creating database and table will be created.
 - c. The code will be executed successfully and create a table but not the database.
 - d. An error is generated without creating database as well as table.
18. Suppose you moved the code `mysql_select_db("my_db", $con);` below the code `$result=mysql_query($sql,$con);`. What would happen to the code after creating the MYSQL connection successfully?
- a. The code will be executed successfully without any error.

St. Mary's University College
Faculty of Informatics

- b. An error is generated in creating database and table will be created.
- c. The code will be executed successfully and create a database but not the table.
- d. An error is generated without creating database as well as table.


19. Write the output of the following code

| | |
|----|---|
| a) | <pre><?php \$txt1="Internet Programming"; \$txt2="Degree Exit Exam"; echo \$txt1 . " " . strlen(\$txt2); ?></pre> |
| b) | <pre><?php \$sentence = "number divide by zero is undefined"; \$words = explode(" ", \$sentence); foreach (\$words as \$dis) print(strlen(substr(\$dis,2))."
"); ?></pre> |
| c) | <pre><?php \$OS = array ("Windows", "Ubuntu", "Fedora", "Solaris", "Mac"); \$list = implode(" ", \$OS); echo \$list; ?></pre> |
| d) | <pre><?php echo "<table border=1 align=center>"; for (\$i = 1; \$i<=5; \$i++){ echo "<tr>"; for (\$j = 1; \$j <=5; \$j++) echo "<td align='center'>" . \$i * \$j . "</td>"; echo "</tr>"; } echo "</table>"; ?></pre> |
| e) | <pre><?php for (\$x=10; \$x<=100; \$x++){ if ((\$x % 19) == 0 && (\$x!=76)) echo "\$x
 "; else continue;} ?></pre> |
| f) | <pre><?php \$email="informatics@smuc.edu.et"; print(strpos(\$email, 'i',5)); ?></pre> |
| g) | <pre><?php \$text="indicator"; print(strtoupper(substr(\$text,4)."
")); print(substr(\$text,-4)."
"); print(substr(\$text,4,3)."
"); print(substr(\$text,4,-3)."
"); ?></pre> |

St. Mary's University College
Faculty of Informatics

Solution for Self Assessment Question

- | | |
|-----------------------------|---------------|
| 1. D | 10. A,C and D |
| 2. A | 11. B |
| 3. D | 12. B |
| 4. B | 13. B |
| 5. C | 14. A |
| 6. C | 15. D |
| 7. A | 16. A |
| 8. B | 17. D |
| 9. C | 18. C |
| 19. The output for the code | |

| | |
|----|--|
| a. | Internet Programming 16 |
| b. | 4
4
0
2
0
7 |
| c. | Windows , Ubuntu , Fedora , Solaris , Mac |
| d. |  |
| e. | 19
38
57
95 |
| f. | 8 |
| g. | CATOR
ator
cat
ca |

Unit self-test question list without answer key

1. Write the output of the following script?

| Code | Output |
|---|--------|
| <pre><?php \$array=array(1,2,3,5,8,13,21,34,55); \$sum=0; for(\$i=0;\$i<2;\$i++) \$sum+=\$array[\$array[\$array[\$i]]]; print(\$sum); ?></pre> | |
| <pre><?php \$array=array(2=>8,3=>27,4=>64,5=>125); foreach (\$array as \$key=>\$value) print(\$key.", ".\$value."
") print(\$sum); ?></pre> | |

2. Write a PHP code that counts the total number of user who have logged in to the system. (Hint use a session control to count and store to DB mechanism).
3. Write a PHP program that generates an ID for new user.
- Assumption :** Read the a column called “Account_ID” from the table “account” after sorting in descending order and do increment the ID by one and generate ID
4. Using conditional statement and iteration control structure to output all the odd integers from 1 to 101 that are not divisible by either 3 or 5.(**use continue and break**)
- Your program should output integers such as 1, 7, 11, 13, 19 ...
5. Write a PHP program that accept an integer value from the form and post to the server side and display the factorial of that number in a message box (Use JavaScript inside PHP).
- factorial (n) = 1 if n < 2, n * (n - 1) * ... * 1 otherwise
6. Write a program that defines two floating point variables. The first variable that represent the monthly income (salary) and the second variable for payable allowance of the employee. The question is to write a PHP program that accept the salary along with the allowance from the form

St. Mary's University College
Faculty of Informatics

then post to the page "Calculate_Salary_Info.php" and finally calculate the amount of tax and net income based on the following condition

| | |
|------------|---------------------|
| Sal<=15 | tax= Sal *0% |
| Sal <=650 | tax= Sal *10%-15 |
| Sal <=1400 | tax= Sal *0.15-47.5 |
| Sal <=2350 | tax= Sal *0.2-117.5 |
| Sal <=3550 | tax= Sal *25%-235 |
| Sal <=5000 | tax= Sal *30%-412.5 |
| Sal >5000 | tax= Sal *35%-662.5 |

Total_income = Sal + allowance, Net_income = Sal +allowance - tax

For the above question allow the user to enter the number of employee using JavaScript which is embedded in PHP and finally display the average, maximum, and minimum statistics about employee salary, allowance and tax in tabular way for the entire employee.

7. Create a PHP class having the attribute machine_name, Username, password and DBName with all setter, getter method along with a constructor and two public method that can do the following:
 - a. Accept SQL statement and execute the query for Insert, Update and Delete.
 - b. Accept select SQL statement and execute the query and finally return the record set to the caller method.
 - c. Create an account form that can post all the information received from the user in to the database table called "Account".

Create New Account ID

| | |
|---|--|
| Account ID | <input type="text"/> |
| Username | <input type="text"/> |
| Password | <input type="password"/> |
| Privilege | <input type="button" value="Select Privilege ▼"/> |
| Account Status | <input type="button" value="Select Account Status ▼"/> |
| <input type="button" value="Create Account"/> | |

St. Mary's University College
Faculty of Informatics

- d. Write a full php program that accept the username and password from the user as shown figure below and check if the username and password exist. If the username and password exist update/change the old password by the new one.

Change Password

| | |
|----------------------|--------------------------|
| Username | <input type="text"/> |
| Old Password | <input type="password"/> |
| New Password | <input type="password"/> |
| Confirm New Password | <input type="password"/> |

8. Write PHP programs that accept a series 0s and 1s (i.e., a binary integer) from the user and display the equivalent Octal and Hexadecimal and display the final result for user screen after rejecting other inputs.
9. Write an application that displays a table of the binary, octal and hexadecimal equivalents of the decimal numbers in the range 1 through 256.
10. An integer is said to be prime if it is divisible by only 1 and the number itself. The question is to write a PHP program that accepts a positive integer and counts the number of primes which are less than the given number entered by the user. Display the count value of the primes less than the number in table.

St. Mary's University
Faculty of Informatics

Study Guide

For

Computer Network

Department of Computer Science
Faculty of Informatics



ቅድስት ማርያም ዩኒቨርሲቲ
St. Mary's University, Ethiopia

Prepared by

Kalechristos Abebe

January 2012

Module summary

The study guide of computer network is organized into seven units, with each unit having its own set of self test problems. The units are selected as, Introduction to data communication and networking, network models, transmission media, LAN technologies and Multiple access, WAN technologies and networks security.

For each of the chapters, self test problems are available as those having answer key and those without answer. All the questions are set to address main areas of the unit mentioned in the specifics objectives list. Students are advised to try to answer all the questions and check out answer for those questions with answer key

Questions listed as those without answer key are all subjective types which help the students to have better understanding of the subject

Unit I

Introduction to data communication and Network

Summary

Data communications are the transfer of data from one device to another via some form of transmission medium. A data communications system must transmit data to the correct destination in an accurate and timely manner. The five components that make up a data communications system are the message, sender, receiver, medium, and protocol. Text, numbers, images, audio, and video are different forms of information. Data flow between two devices can occur in one of three ways: simplex, half-duplex, or full-duplex.

A network is a set of communication devices connected by media links. In a point-to-point connection, two and only two devices are connected by a dedicated link. In a multipoint connection, three or more devices share a link. Topology refers to the physical or logical arrangement of a network. Devices may be arranged in a mesh, star, bus, or ring topology. A network can be categorized as a local area network or a wide area network. A LAN is a data communication system within a building, plant, or campus, or between nearby buildings. A WAN is a data communication system spanning states, countries, or the whole world. An internet is a network of networks. The Internet is a collection of many separate networks. There are local, regional, national, and international Internet service providers. A protocol is a set of rules that govern data communication; the key elements of a protocol are syntax, semantics, and timing. Standards are necessary to ensure that products from different manufacturers can work together as expected. The ISO, ITD-T, ANSI, IEEE, and EIA are some of the organizations involved in standards creation. Forums are special-interest groups that quickly evaluate and standardize new technologies. A Request for Comment is an idea or concept that is a precursor to an Internet standard.

General objective

The general objective of the unit is to help students understand the basic concepts of data communication and computer networks

Specific Objectives

Specific objectives of the unit include:

- Define data communication
 - Identify Components of data communication system
 - Identify Characteristics of data communication system
 - Describe direction of data flow
- Identify and describe computer networks
 - Describe Network criteria
 - Describe uses of computer network
 - Explain network physical structure
 - Identify Types of connection
 - Identify Physical topology
 - ✓ Explain the physical layout of topologies
 - ✓ Explain advantages and disadvantages of topologies
 - Identify categories of networks
 - Identify Local Area network(LAN)
 - Identify Wide Area network(WAN)
 - Identify Metropolitan Area Network(MAN)
- Identify the Internet
 - Explain the brief history of Internet
 - Describe Internet service providers
- Identify Protocol and Standards
 - Identify protocols
 - Identify standards
 - Identify standard organizations

Self test Problems without answer key

1. What are the advantages of a multipoint connection over a point-to-point connection?
2. Categorize the four basic topologies in terms of their line configuration
3. Suppose you add two new devices to an existing five device network. If you have a fully connected mesh topology, how many new cable lines are needed? If devices are arranged in a single ring, how many new cable lines are needed?
4. Draw a hybrid topology, with bus back bone connecting two ring back bones. Each ring backbone connects three star networks
5. What are some factors that determine whether a communication system is LAN, MAN or WAN.
6. What is an internet? What is an Internet?
7. Why are protocols and standards needed?

Self test Problems with answer key

Multiple choices

1. Frequency of failure and network recovery time after failure are measures of _____ of a network
 - a. Performance
 - b. Reliability
 - c. Security
 - d. Feasibility
2. Which topology requires a central controller or hub?
 - a. Star
 - b. Mesh
 - d. Ring
 - Bus
3. The _____ is a physical path over which a message travels
 - a. Protocol
 - b. Medium
 - c. Signal
 - d. Standard
4. Which topology requires multipoint connection?
 - a. Mesh
 - b. Star
 - c. Bus
 - d. Ring
5. Communication between computer and keyboard involves _____
 - a. Simplex
 - b. Duplex
 - c. Half duplex
 - d. Automatic
6. A _____ connection provides a dedicated link between two devices

St. Mary's University
Faculty of Informatics

- a. Point-to-point b. Multi point c. Primary d. Secondary
7. A cable break in _____ topology stops all transmission
a. Mesh b. Bus c. Star d. Ring
8. In a network with 25 computers, which topology requires most extensive cabling?
a. Mesh b. Star c. Bus d. Ring
9. A publishing company with head quarters in Addis Abeba and branch offices throughout Africa, Europe and Asia is probably connected by _____
a. LAN b. WAN c. MAN d. None of the above
10. In _____ communication, the channel capacity is used by both communicating devices at all times
a. Simplex b. Half-duplex c. Full-duplex d. Half-simplex

Answer the following questions

11. Identify the five components a data communication system
12. What are the advantages of a multipoint connection over a point-to-point connection
13. Name the four basic network topologies, and cite an advantage of each type
14. Assume six devices are arranged in a mesh topology, how many cables are needed? How many ports are needed for each device?
15. What are the three criteria needed for an efficient and effective network?

Answer Key

- | | | |
|--------------|-------------|-------------|
| 1. B | 2. A | 3. B |
| 4. C | 5. A | 6. A |
| 7. D | 8. A | 9. B |
| 10. C | | |

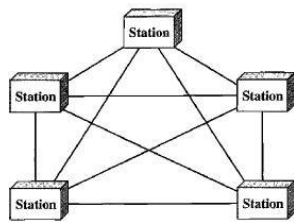
11. The five components of a data communication system are:
- I. Message: information (data) to be communicated
 - II. Sender: sender is a device that sends a data message

- III. Receiver: Receiver is a device that receives the message
- IV. Medium: Transmission medium is a physical path by which message travels for sender to receiver
- V. Protocol: Protocol is a set of rules that governs data communication

12. The advantage of multi point connection over point to point connection is that multi point connection provides greater efficiency since more than two devices could share the link

13. The four basic topologies with their advantages and disadvantages

Mesh Topology: Each device has a dedicated point to point link to every other device in the network



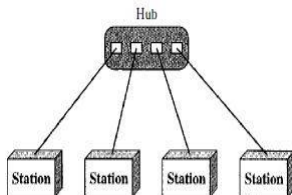
Advantage

- Eliminates traffic problem
- Robust
- Privacy or security

Disadvantages

- Extensive cabling and more number of ports are needed

Star Topology: Each device has a point to point link to a central controller device or hub



Advantage

- Less expensive
- Easy to install and configure
- Robust

Bus Topology: In bus topology one cable acts as a back bone to link all other devices in the network



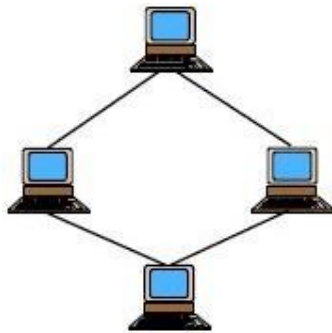
Advantage

- Ease of installation

Disadvantage

- Difficult re connection and fault isolation

Ring Topology: Each device has a dedicated point to point connection with only two devices on either side of it



Advantage

- Easy to install and configure
- Simple fault isolation

Disadvantage

- Unidirectional traffic
- Single point of failure

14. For six devices arranged in a fully connected mesh topology:

$$\text{Number of cables needed} = 6 \times (6-1) / 2 = 15$$

$$\text{Number of ports needed for each device} = 6-1 = 5$$

15. The three criteria's needed for efficient and effective network are@

I. Performance :Performance can be measured with transit time and response time of the network

II. Reliability: is a measure network frequency of failure, time it takes to recover from failure, and network's robustness in catastrophe

III. Security: Security issues include protecting data for un authorized access

Unit Two

Network Models

Summary

The International Standards Organization created a model called the Open Systems Interconnection, which allows diverse systems to communicate. The seven-layer OSI model provides guidelines for the development of universally compatible networking protocols. The physical, data link, and network layers are the network support layers. The session, presentation, and application layers are the user support layers

TCP/IP is a five-layer hierarchical protocol suite developed before the OSI model. The TCP/IP application layer is equivalent to the combined session, presentation, and application layers of the OSI model. Four levels of addresses are used in an internet following the TCP/IP protocols: physical (link) addresses, logical (IP) addresses, port addresses, and specific addresses. The physical address, also known as the link address, is the address of a node as defined by its LAN or WAN. The IP address uniquely defines a host on the Internet, The port address identifies a process on a host A specific address is a user-friendly address

General Objective

The general objective of the unit is to understand the network reference model

Specific Objectives

At the end of the unit students are expected to

- Understand the concepts of OSI reference model
 - Identify layered architecture
 - Identify peer-to-peer process
 - Identify encapsulation
 - Identify layers in the OSI model
 - Explain the function of Physical layer
 - Explain the function of data link layer
 - Explain the function of network layer
 - Explain the function of transport layer
 - Explain the function of session layer
 - Explain the function of presentation layer
 - Explain the function of application layer
- Understand the concepts of TCP/IP reference model
 - Identify TCP/IP layers
 - Identify physical and data link layers
 - Identify network layer
 - Identify transport layer
 - Identify application layer
 - Identify TCP/IP protocols
 - Identify Internet protocol
 - Identify Transmission control protocol
 - Identify user datagram protocol
 - identify application layer protocols
 - identify IP addressing techniques
 - Explain IP address assignments
 - Explain IP address classes
 - Identify subnet addressing

- Explain subnetting
- Explain subnet masking

Self test problems without answer key

1. What are the advantages of using UDP over TCP?
2. Why is domain name reversed when searching for the IP address?
3. Write the functions of the following application layer protocols
 - i. DHCP
 - ii. DNS
 - iii. FTP
 - iv. RPC
4. Draw two Ethernet LANs connected by a gateway. Each LAN has three hosts. One LAN is class B and one is class C. Choose appropriate internet addresses. How many connections must the gateway have?
5. What is the difference between physical address and logical address?
6. Find the classes of the following IP addresses
 - a. 121.56.3.67
 - b. 193.23.56.23
 - c. 231.23.67.123
 - d. 142.23.56.23
7. Find the network addresses for IP addresses given in problem 6
8. Give some advantages and disadvantages of combining the session, presentation, and application layer in the OSI model into one single application layer as in the Internet model
9. For the given class B IP address give below, find
IP:255.255.254.0
 - i. Network address
 - ii. Subnet Mask
 - iii. Broad cast address

Self test problem with answer key

Choose the best answer for the following

1. The_____ layer of OSI model decides the location of synchronization points
 - a. Transport
 - b. Session
 - c. Presentation
 - d. Application
2. Which of the following layers of OSI model is responsible for end to end delivery of entire message?
 - a. Network
 - b. Transport
 - c. Session
 - d. Presentation
3. In the _____layer of OSI model , data unit is called a frame
 - a. Physical
 - b. Data link
 - c. Network
 - d. Transport
4. Which layer of the OSI model uses the trailer of the frame for error detection?
 - a. Physical
 - b. Data link
 - c. Transport
 - d. Presentation
5. On which layer of the OSI model both header and trailer information are added?
 - a. Physical
 - b. Network
 - c. Data link
 - d. Transport
6. Which of the following applies of UDP?
 - a. Un reliable and connectionless
 - b. Contains destination and source port addresses
 - c. Reports certain errors
 - d. All of the above
7. Which of the following is a class A network address?
 - a. 128.4.5.6
 - b. 127.4.5.0
 - c. 127.0.0.0
 - d. 127.8.0.0
8. Which of the following is a class B host address
 - a. 230.0.0.0
 - b. 130.4.5.6
 - c. 230.4.5.9
 - d. 30.4.5.6
9. Which IP address class has few hosts per network?

St. Mary's University College

Faculty of Informatics

- a. A b. B c. C d. D

10. The data unit in TCP/IP data link layer is called _____

- a. Message b. Segment c. Datagram d. Frame

Answer the following questions

11. Describe the fields available in IPV6 header format

12. Physical, port, and IP addresses are used in data communication. In TCP/IP environment, what layers are they associated with?

13. What does subnetting and subnet masking refers to?

14. For the given class B IP address give below, find

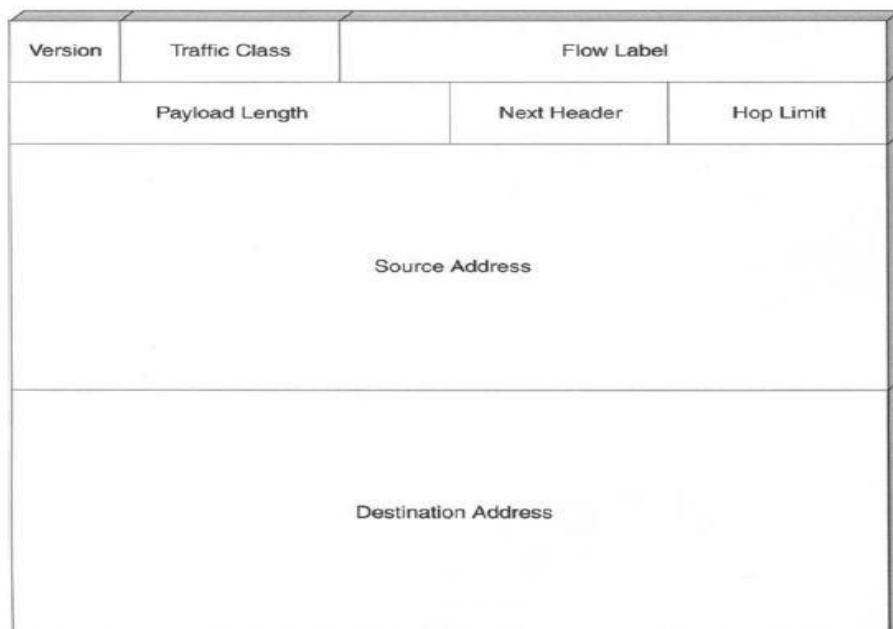
IP:255.255.192.0

- i. Network address
- ii. Subnet Mask
- iii. Broad cast address

Answer Key

- | | | |
|------|------|-------|
| 1. B | 2. B | 3. B |
| 4. B | 5. C | 6. D |
| 7. C | 8. B | 9. C |
| | | 10. D |

11. IPV6 Header format is as follows



Traffic Class: This 8-bit value specifies what, if any, form of differentiated service is to be provided for the packet. Use of this field is defined separately from IPv6; see RFC 2474 for more about differentiated services. The default value for this field is all zeros.

Flow label: This 20-bit value identifies packets that belong to the same flow. A node can be the source for more than one simultaneous flow. The flow label and the address of the source node uniquely identify flows.

Payload length: This 16-bit field contains an integer value equal to the length of the packet payload in octets; that is, the number of octets contained in the packet after the end of the IPv6 header. IPv6 extensions are included as part of the payload for the purposes of calculating this field.

Next header: This field indicates what protocol is in use in the header immediately following the IPv6 packet. Similar to the IPv4 protocol field, the next header field may refer to a higher layer protocol like TCP or UDP, but it may also indicate an IPv6 extension header.

Hop limit: Every time a node forwards a packet, it decrements this eight-bit field by one. If the hop limit reaches zero, the packet is discarded. Unlike in IPv4, where the time-to-live field fulfills a similar purpose, sentiment is currently against putting a protocol-defined upper limit on packet lifetime for IPv6. This means that the function of timing-out old data should be accomplished in upper layer protocols.

Destination address: This is the 128-bit address of the intended recipient of the IPv6 packet. This address may be a unicast, multicast, or anycast address. If a routing extension is being used (which specifies a particular route that the packet must traverse), the destination address may be one of those intermediate nodes instead of the ultimate destination node.

12. The

physica

l

address is

associated with

data link layer. The logical address is associated with the network layer. A port address is

associated with application layer

the

St. Mary's University

Faculty of Informatics

13. Subnetting refers to the partitioning of a network address space into separate autonomous sub networks.

14. IP address: **255.255.192.0**

172.16.0.0=Network address

255.255.192.0=Subnet mask

1. $2^2 - 2 = 2$.

2. $2^{14} - 2 = 16,382$.

3. $256 - 192 = 64$. $64 + 64 = 128$.

4. First find the broadcast addresses in step 5, then come back and perform step 4 by filling in the host addresses.

5. Find the broadcast address of each subnet, which is always the number right before the next subnet.

The following table shows the two subnets available, the valid host range, and the broadcast address of each.

| | | |
|------------|---------|---------|
| Subnet | 64.0 | 128.0 |
| First Host | 64.1 | 128.1 |
| Last Host | 127.254 | 191.254 |
| Broadcast | 127.255 | 191.255 |

Notice we just added the fourth octet's lowest and highest values and came up with the answers. Again, it is the same answer as for a Class C subnet, but we just added the fourth octet.

Unit 3

Transmission Media

Summary

A transmission **medium** can be broadly defined as anything that can carry information from a source to a destination. Transmission media lie below the physical layer. Guided medium provides a physical conduit from one device to another. Twisted pair cable, coaxial cable, and optical fiber are the most popular types of guided media. Twisted-pair cable consists of two insulated copper wires twisted together. Twisted pair cable is used for voice and data communications. Coaxial cable consists of a central conductor and a shield. Coaxial cable can carry signals of higher frequency ranges than twisted-pair cable. Coaxial cable is used in cable TV networks and traditional Ethernet LANs. Fiber-optic cables are composed of a glass or plastic inner core surrounded by cladding, all encased in an outside jacket. Fiber-optic cables carry data signals in the form of light. The signal is propagated along the inner core by reflection. Fiberoptic transmission is becoming increasingly popular due to its noise resistance, low attenuation, and high-bandwidth capabilities. Fiber-optic cable is used in backbone networks, cable TV networks, and Fast Ethernet networks. Unguided media (free space) transport electromagnetic waves without the use of a physical conductor.

Wireless data are transmitted through ground propagation, sky propagation, and line of-sight propagation. Wireless waves can be classified as radio waves, microwaves, or infrared waves. Radio waves are Omni-directional; microwaves are unidirectional. Microwaves are used for cellular phone, satellite wireless LAN communications. Infrared waves are used for short-range communications such as those between a PC and a peripheral device. It can also be used for indoor LANs.

General Objective

The general objective of the unit is to understand use of transmission media at the physical layer

Specific Objectives

At the end of the unit, students are expected to:

- Understand use of guide media

St. Mary's University

Faculty of Informatics

- Explain the working mechanism of twisted pair cable
- Explain the working mechanism of coaxial cable
- Explain the working mechanism of Fiber optic cable
- Understand the use of Un guided transmission media
 - Explain the working mechanism of Radio waves
 - Explain the working mechanism of microwaves
 - Explain the working mechanism of infrared

Self test problems without answer key

1. How do guided media differ from unguided media?
2. How does sky propagation differ from line of sight propagation?
3. Differentiate straight through and cable cross over connections in twisted pair
4. Name the advantages of fiber optic over twisted pair cable
5. Differentiae among the following coaxial cable standards
 - i. RG-8
 - ii.RG-9 iii
 - RG-11
 - iv RG-58

Self test problems with answer key

Choose the best answer for the following

1. Transmission media are usually categorized as _____
 - a. Fixed or unfixed
 - b. Guided or unguided
 - c. Determined or undetermined
 - d. Metallic or non metallic
2. In fiber optics , the signal source is _____waves
 - a. Light
 - b. Radio

- c. Infrared
 - d. Very low frequency
3. In an environment with many high voltage devices , the best transmission medium is____
- a. Twisted pair cable
 - b. Coaxial cable
 - c. Optical fiber cable
 - d. Atmospheres
4. What is the reason that makes coaxial cable less susceptible to noise than twisted pair cable?
- a. Inner conductor
 - b. Diameter of cable
 - c. Outer conductor
 - d. Insulating material
5. The inner core of an optical fiber is _____in composition
- a. Glass or plastic
 - b. Copper
 - c. Bimetallic
 - d. Liquid

Answer the following questions

- 6. Explain what cross talk is and what is needed to reduce it
- 7. Why should the light ray be reflective rather than refractive in fiber optics?
- 8. Why is there a distance limit of terrestrial microwaves? What factors do you need to calculate this limit?
- 9. Discuss the different categories of twisted pair cable

Answer Key

- | | | |
|------|------|------|
| 1. B | 2. A | 3. C |
| 4 C | 5. A | |

St. Mary's University

Faculty of Informatics

6. Cross talk is the undesired effect of one circuit (channel) on another circuit. Cross talk can be reduced by shielding each pair of twisted pair cable
7. In fiber optics cable the light ray needs to travel along a long narrow channel, the core; it does this by successive reflections against the cladding. On the other hand , a refractive ray goes into the cladding and leaves the core area; the information thus does not get propagated; it is lost
8. Terrestrial microwave transmission is limited by the curvature of the earth and the height of the antenna. Both of these factors are needed to calculate the distance a signal can travel
9. The different categories of twisted pair cable are shown in the table below

| <i>Category</i> | <i>Specification</i> | <i>Data Rate (Mbps)</i> | <i>Use</i> |
|-----------------|---|-------------------------|------------|
| 1 | Unshielded twisted-pair used in telephone | < 0.1 | Telephone |
| 2 | Unshielded twisted-pair originally used in T-lines | 2 | T-IIines |
| 3 | Improved CAT 2 used in LANs | 10 | LANs |
| 4 | Improved CAT 3 used in Token Ring networks | 20 | LANs |
| 5 | Cable wire is normally 24 AWG with a jacket and outside sheath | 100 | LANs |
| SE | An extension to category 5 that includes extra features to minimize the crosstalk and electromagnetic interference | 125 | LANs |
| 6 | A new category with matched components coming from the same manufacturer. The cable must be tested at a 200-Mbps data rate. | 200 | LANs |
| 7 | Sometimes called SSTP (shielded screen twisted-pair). Each pair is individually wrapped in a helical metallic foil followed by a metallic foil shield in addition to the outside sheath. The shield decreases the effect of crosstalk; and increases the data rate. | 600 | LANs |

Unit 4

Local Area Networks and multiple Access

Summary

We can consider the data link layer as two sub layers. The upper sublayer is responsible for data link control, and the lower sub layer is responsible for resolving access to the shared media. Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups: random access protocols, controlled access protocols, and channelization protocols.

Although a LAN can be used as an isolated network to connect computers in an organization for the sole purpose of sharing resources, most LANs today are also linked to a wide area network (WAN) or the Internet. Ethernet is the most widely used local area network protocol. The IEEE 802.3 Standard defines I-persistent *CSMA/CD* as the access method for first-generation 10-Mbps Ethernet. In this unit, the focus is on IEEE Standard Project 802, designed to regulate the manufacturing and interconnectivity between different LANs

General objective

The general objective of the unit is to understand different LAN technologies and multiple access schemes

Specific objectives

At the end of the unit students are expected to:

- Define multiple access
 - Explain random access techniques
 - Explain how carrier sense multiple access (CSMA) works
 - Explain how carrier sense multiple access with collision detection (CSMA/CD) works
 - Explain how carrier sense multiple access with collision avoidance (CSMA/CA) works
 - Explain controlled access techniques
 - Explain how reservation technique works
 - Explain how polling technique works
 - Explain how token passing technique works

- Explain channelization techniques
 - Explain how Frequency –Division multiple access(FDMA) works
 - Explain how Time-Division Multiple access(TDMA) works
 - Explain how Code-Division Multiple access (CDMA) works
- Define LAN technologies
 - Explain the IEEE standards
 - Explain functions of data link layer
 - Explain functions of physical layer
 - Explain the standard Ethernet
 - Explain Ethernet topologies
 - Explain the functions of MAC sub layer
 - Explain bridged Ethernet
 - Explain switched Ethernet
 - Explain full-duplex Ethernet
 - Explain fast Ethernet
 - Explain gigabit Ethernet

Self test problems without answer key

1. Define random access and list protocols in this category
2. Explain why collision is an issue in a random access protocol but not in controlled access or channelizing protocols
3. Compare and contrast a controlled access protocol with a channelizing protocol.
4. Write the functions of the following data link sub layers
 - a. LLC
 - b. MAC
5. Differentiate among the following types of Ethernet
 - a. Bridged Ethernet
 - b. Switched Ethernet
 - c. Giga bit Ethernet

St. Mary's University

Faculty of Informatics

6. Compare the following LAN technologies in terms of their topology, alternate topology, access method and transmission media used
- a. Ethernet
 - b. Token ring

Self test problems with answer key

Choose the best answer for the following

1. In CSMA/CD, the number of collision is _____ than MA
 - a. Greater than
 - b. Less than
 - c. Equal to
 - d. twice
2. In Ethernet , the source address field in the MAC frame is _____ address
 - a. The original senders physical
 - b. The previous stations physical
 - c. The next destinations physical
 - d. The original senders service port
3. Which of the following use star topology?
 - a. 10BASE5
 - b. 10BASE2
 - c. 10BASE-T
 - d. None of the above
4. 10BASE2 uses _____ cable while 10BASE5 uses _____
 - a. Thick coaxial , thin coaxial
 - b. Twisted pair , thick coaxial
 - c. Thin coaxial , thick coaxial
 - d. Fiber optic, thin coaxial
5. _____ specifies a star topology featuring central hub and daisy chaining
 - a. 10BASE5
 - b. 10BASE2
 - c. 10BASE-T
 - d. 1BASE5
6. What can happen at a token ring station?
 - a. Examination of destination address
 - b. Regeneration of the frame

- c. Passing of the frame to the next station
 - d. All of the above
7. In token ring , where is the token when a data frame is in circulation?
- a. At the receiving station
 - b. At the sending station
 - c. Circulating the ring
 - d. None of the above
8. Which LAN has the highest data rate?
- a. 10BASE5
 - b. 10BASE-T
 - c. Twisted –pair token ring
 - d. FDDI
9. In which OSI layer does the FDDI protocol operate?
- a. Physical
 - b. Data link
 - c. Network
 - d. A and b
10. Which project of 802 standard provides for collision-free protocols?
- a. 802.2
 - b. 802.3
 - c. 802.5
 - d. 802.6

Answer the following questions

- 11. Explain how CSMA/CD works
- 12. Explain how token passing works
- 13. Explain the how the FDMA channel access method works

Answer Key

- | | | |
|------|------|-------|
| 1. B | 5. D | 9. D |
| 2. B | 6. D | 10. C |
| 3. C | 7. B | |
| 4. C | 8. D | |

11. The basic idea behind *CSMA/CD* is that a station needs to be able to receive while transmitting to detect a collision. When there is no collision, the station receives one signal: its own signal. When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station. To distinguish between these two cases, the received signals in these two cases must be significantly different. In other words, the signal from the second station needs to add a significant amount of energy to the one created by the first station.
12. In the token-passing method, the stations in a network are organized in a logical ring. In other words, for each station, there is a *predecessor* and a *successor*. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring. The current station is the one that is accessing the channel now. The right to this access has been passed from the predecessor to the current station. The right will be passed to the successor when the current station has no more data to send
13. In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a band pass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small *guard bands*. FDMA specifies a predetermined frequency band for the entire

period of communication. This means that stream data (a continuous flow of data that may not be packetized) can easily be used with FDMA

Unit 5

Connecting LAN's, Backbone networks, virtual LAN's

Summary

LANs do not normally operate **in** isolation. They are connected to one another or to the Internet. To connect LANs, or segments of LANs, we use connecting devices. Connecting devices can operate **in** different layers of the Internet model.

A repeater is a connecting device that operates in the physical layer of the Internet model. A repeater regenerates a signal, connects segments of a LAN, and has no filtering capability. A bridge is a connecting device that operates in the physical and data link layers of the Internet model. A transparent bridge can forward and filter frames and automatically build its forwarding table. A bridge can use the spanning tree algorithm to create a loop less topology. A backbone LAN allows several LANs to be connected. A backbone is usually a bus or a star. A virtual local area network (VLAN) is configured by software, not by physical wiring. Membership in a VLAN can be based on port numbers, MAC addresses, IP addresses, IP multicast addresses, or a combination of these features. VLANs are cost- and time-efficient, can reduce network traffic, and provide an extra measure of security.

General Objective

To understand the different ways of connecting LAN's, backbone network and VLAN's

Specific Objectives

At end of the unit students are expected to:

- Understand LAN connecting devices
 - Explain the operations of Hubs

- Describe Hub operation
- Describe Hub features
- Describe Peer versus stand alone hubs
- Describe intelligent Hubs
- Describe multi architecture Hubs
- Explain operations of Switches
 - Describe operations of switches
 - Describe two layer switches
 - Describe three layer switches
- Explain the operations of repeaters
 - Describe repeaters and network architecture
 - Describe repeater to repeater connection
- Explain the operations of routers
 - Describe Route Discovery
 - Describe Router Operation
 - Describe Router Groupings
 - Describe Router Protocols
- Explain operations of bridges
 - Bridges versus Routers, Brouters, and Repeaters
 - Protocol Independence of Bridges
 - Packet Transmission
 - Types of Bridges
- Explain the operations of gateways
 - Gateways in Networks
 - Gateway Operation
 - Gateway Categories
- Understand backbone networks

St. Mary's University

Faculty of Informatics

- Explain bus backbone network
- Explain star backbone network
- Explain connecting remote LANs
- Understand virtual LAN
 - Describe VLAN membership
 - Describe VLAN configuration

Self test problems without answer key

1. Differentiate among the following types of LAN switches
 - i. Store and forward
 - ii. Cut through
2. Describe the working mechanism of ISL protocol for VLAN
3. How does VLAN reduce network traffic?
4. What do we mean when we say that a bridge can filter traffic? Why is filtering important?
5. Write the operations of distance vector and link state routing algorithms
6. How is a repeater different from amplifier
7. Describe operations of bus and star backbone networks

Self test problems with answer key

Choose the best answer for the following

1. Which of the following uses greater number of layers in OSI model?
 - a. Bridge
 - b. Repeater
 - c. Router
 - d. Gateway

St. Mary's University

Faculty of Informatics

2. A bridge forwards or filters a packet comparing the information in its address table to the packet's _____
 - a. Layer 2 source address
 - b. Source node's physical address
 - c. Layer 2 destination address
 - d. Layer 3 destination address
3. A simple bridge does which of the following?
 - a. Filters a data packet
 - b. Forwards a data packet
 - c. Extends LAN's
 - d. All of the above
4. Which of the following are bridge types?
 - a. Simple, complex, learning
 - b. Simple, learning, multiport
 - c. Simple, complex, multiport
 - d. Spanning, contract, suspension
5. The shortest path in routing refers to _____
 - a. The least expensive path
 - b. The least distant path
 - c. The path with smallest number of hops
 - d. Any or combination of then above
6. Gateways function in which OSI layers?
 - a. The lower three
 - b. The upper four
 - c. All seven
 - d. All but the physical layer
7. A repeater takes a weakened signal and _____ it

- a. Amplifies
 - b. Regenerates
 - c. Resamples
 - d. Reroutes
8. In distance vector routing each router receives vectors from_____
- a. Every router in the network
 - b. Every router less than two units away
 - c. A table stored by the software
 - d. It's neighbours only
9. In link state routing , flooding allows changes to be recorded by _____
- a. All routers
 - b. Neighbour routers only
 - c. Some routers
 - d. All networks
10. For which of the following would you not need to provide a crossover cable?
- a. Connecting uplinks between switches
 - b. Connecting routers to switches
 - c. Connecting hub to hub
 - d. Connecting hub to hub

Answer the following questions

- 11. Write the three distinct functions of layer2 switching
- 12. Describe the difference between static and dynamic VLAN
- 13. Describe the active and passive hubs

Answer key

- | | | |
|------|------|------|
| 1. D | 3. D | 5. D |
| 2. C | 4. B | 6. C |

7. B

9. A

8. D

10. B

11. There are three distinct functions of layer-2 switching:

Address learning:

Layer-2 switches and bridges remember the source hardware address of each frame received on an interface and enter this information into a MAC database.

Forward/filter decisions: When a frame is received on an interface, the switch looks at the destination hardware address and finds the exit interface in the MAC database.

Loop avoidance: If multiple connections between switches are created for redundancy, network loops can occur. The Spanning-Tree Protocol (STP) is used to stop network loops and allow redundancy.

12. *Static VLANs* are the typical way of creating VLANs and the most secure. The switch port that you assign a VLAN association always maintains that association until an administrator changes the port assignment. This type of VLAN configuration is easy to set up and monitor, working well in a network where the movement of users within the network is controlled. Using network management software to configure the ports can be helpful but is not mandatory.

Dynamic VLANs determine a node's VLAN assignment automatically. Using intelligent management software, you can enable hardware (MAC) addresses, protocols, or even applications to create dynamic VLANs. For example, suppose MAC addresses have been entered into a centralized VLAN management application. If a node is then attached to an unassigned switch port, the VLAN management database can look up the hardware address and assign and configure the switch port to the correct VLAN. This can make management and configuration easier for the

administrator. If a user moves, the switch will automatically assign them to the correct VLAN. However, more administration is needed initially to set up the database.

13. Passive Hubs

A passive hub is just a connector. It connects the wires coming from different branches. In a star-topology Ethernet LAN, a passive hub is just a point where the signals coming from different stations collide; the hub is the collision point. This type of a hub is part of the media; its location in the Internet model is below the physical layer.

Active Hubs

An active hub is actually a multipart repeater. It is normally used to create connections between stations in a physical star topology. We have seen examples of hubs in some Ethernet implementations (10Base-T, for example). However, hubs can also be used to create multiple levels of hierarchy

Unit 6

Wide Area Networks

Summary

To understand WAN technologies, you need to understand the different WAN terms and connection types that can be used to connect your networks together. This unit focuses the different WAN terms and connection types typically used by service providers. Leased line, circuit switching and packet switching are types of connections to be used. The unit also focuses on different WAN technologies like , Distributed Queue Dual Bus (DQDB) ,Synchronous Digital Hierarchy (SDH),Synchronous Optical Network (SONET) Asynchronous Transfer Mode (ATM)

General Objective:

To understand different WAN connections and technologies

Specific Objectives:

At the end of the unit students are expected to

- Understand the concept of WAN
 - Describe WAN connection types
 - Describe leased line connection
 - Describe circuit switching
 - Describe packet switching

Describe WAN technologies

- Describe Synchronous Digital Hierarchy (SDH)
- Describe Synchronous Optical Network (SONET)
- Describe Asynchronous Transfer Mode (ATM)
- Frame relay
- Describe integrated services digital network(ISDN)

Self test problems without answer key

1. What is the relationship between SONET and SDH?
2. There are no sequence numbers in Frame Relay. Why?
3. How is an ATM virtual connection identified?
4. Briefly describe the issues involved in using ATM technology in LANs.
5. Discuss the Frame Relay physical layer.

Self test problems with answer key

Choose the best answer

1. ATM can use_____as transmission medium
 - a. Twisted pair cable
 - b. Coaxial cable
 - c. Fiber optic cable

- d. All of the above
- 2. Which layer in ATM protocol reformats the data received from other networks?
 - a. Physical
 - b. ATM
 - c. Application adaptation
 - d. Data adaptation
- 3. Which AAL type can best process a data stream having a non constant bit rate?
 - a. AAL1
 - b. AAL2
 - c. AAL3/4
 - d. AAL5
- 4. SONET is a standard for_____ networks
 - a. Twisted pair cable
 - b. Coaxial cable
 - c. Ethernet
 - d. Fiber optic cable
- 5. SONET's _____layer correspond to the OSI model's physical layer
 - a. Path
 - b. Line
 - c. Section
 - d. Photonic
- 6. The optical link between two SONET devices is called_____
 - a. Section
 - b. Line
 - c. Path
 - d. None of the above
- 7. What does the ISDN Basic Rate Interface (BRI) provide?

St. Mary's University

Faculty of Informatics

- b. 23 B channels and one 64Kbps D channel
- c. Total bit rate of up to 1.544Mbps
- d. Two 56Kbps B channels and one 64Kbps D channel
- e. Two 64Kbps B channels and one 16Kbps D channel

Answer the following questions

- 8. Describe the leased line, circuit switching and packet switching WAN connection types
- 9. Discuss the functions of each SONET layer
- 10. Name the ATM layers

Answer Key

- | | |
|------|------|
| | 4. D |
| 1. D | 5. C |
| 2. C | 6. A |
| 3. B | 7. B |

- 8. The following are WAN connection types:

Leased lines Typically referred to as a point-to-point or dedicated connection. It is a pre-established WAN communications path from the CPE, through the DCE switch, to the CPE of the remote site, allowing DTE networks to communicate at any time with no setup procedures before transmitting data. It uses synchronous serial lines up to 45Mbps.

Circuit switching

Sets up line like a phone call. No data can transfer before the end-to-end connection is established. Uses dial-up modems and ISDN. It is used for low-bandwidth data transfers.

Packet switching

WAN switching method that allows you to share bandwidth with other companies to save money. Think of packet switching networks as a party line. As long as you are not constantly transmitting data and are instead using bursty data transfers, packet switching

can save you a lot of money. However, if you have constant data transfers, then you will need to get a leased line. Frame Relay and X.25 are packetswitching technologies. Speeds can range from 56Kbps to 2.048Mbps

9. The following are SONET layers and their functions

Path Layer

The path layer is responsible for the movement of a signal from its optical source to its optical destination. At the optical source, the signal is changed from an electronic form into an optical form, multiplexed with other signals, and encapsulated in a frame. At the optical destination, the received frame is demultiplexed, and the individual optical signals are changed back into their electronic forms. Path layer overhead is added at this layer. STS multiplexers provide path layer functions.

Line Layer

The **line layer** is responsible for the movement of a signal across a physical line. Line layer overhead is added to the frame at this layer. STS multiplexers and add/drop multiplexers provide line layer functions.

Section Layer

The **section layer** is responsible for the movement of a signal across a physical section. It handles framing, scrambling, and error control. Section layer overhead is added to the frame at this layer.

Photonic Layer

The **photonic layer** corresponds to the physical layer of the OSI model. It includes physical specifications for the optical fiber channel, the sensitivity of the receiver, multiplexing functions, and so on. SONET uses NRZ encoding with the presence of light representing 1 and the absence of light representing 0

10. The following are ATM layers and their function

- A. Application adaptation layer (AAL)
- B. ATM Layer

Unit 7

Network Security

Summary

The increase in attacks of message during transmission coincides with an increased use of the Internet and with increases in the complexity of protocols, applications, and the Internet itself. Critical infrastructures increasingly rely on the Internet for operations. Individual users rely on the security of the Internet, email, the Web, and Web-based applications to a greater extent than ever. Thus, a wide range of technologies and tools are needed to counter the growing threat. At a basic level, cryptographic algorithms for confidentiality and authentication assume greater importance. As well, designers need to focus on Internet-based protocols and the vulnerabilities of attached operating systems and applications.

Thus network security typically deals applying different cryptographic techniques in order secure a message selecting different protocols and algorithms to counter measure attacks

General objective

To understand the different network security techniques

Specific Objectives

At the end of this unit students are expected to :

- Describe Security Trends
- Explain The OSI Security Architecture
- Explain Security Attacks

St. Mary's University

Faculty of Informatics

- Explain Passive Attacks and Active Attacks
- Describe Security Services
 - Authentication
 - Access Control
 - Data Confidentiality
 - Data Integrity
 - Nonrepudiation
 - Availability Service
 - Security Mechanisms
- Describe Confidentiality with Symmetric Encryption
 - Explain Symmetric Encryption
 - Explain Encryption Algorithms
 - Explain Location of Encryption Devices
- Describe Message Authentication and Hash Functions
- Describe Public-Key Encryption and Digital Signature
 - Explain Public Key Encryption
 - Explain Digital Signature
 - Explain The RSA Public-Key Encryption Algorithm
 - Explain Key Management

- Describe Secure Socket Layer and Transport Layer Security
 - Explain SSL Architecture
- Describe IP security and email security
 - Explain IPV4 and IPV6 securities
 - Explain the scope of IPSec

Self test problems without answer key

1. Differentiate between security services and mechanisms
2. Define the following terms
 - a. Cryptography
 - b. Cryptanalysis
3. What are important features of symmetric encryption
4. Describe the hash function properties and procedures
5. Differentiate between public and private keys
6. Explain the key management techniques used in public crypto system
7. Write the digital signature algorithm and its properties
8. Explain the SSL architecture

Self test problems with answer key

1. Differentiate between active and passive attacks
2. Describe the meaning of the following security services
 - a. Access control
 - b. Non repudiation
 - c. Authentication
3. Define the following basic terminologies
 - a. Plain text
 - b. Cipher text
 - c. Cipher
 - d. Key
 - e. Encryption
 - f. Decryption
4. Explain how cryptographic systems are characterized
5. Describe the message authentication code (MAC) procedures
6. Write essential differences of symmetric(conventional) and asymmetric(public key) encryption
7. Write the RSA public key encryption/decryption algorithm

8. What are IPSec services?
9. Write the PGP operations used for email security

Answer key

1. Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions.
The goal of the opponent is to obtain information that is being transmitted

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service

2. **Access Control** - prevention of the unauthorized use of a resource

Non-Repudiation - protection against denial by one of the parties in a in

Communication

Authentication - assurance that the communicating entity is the one claimed

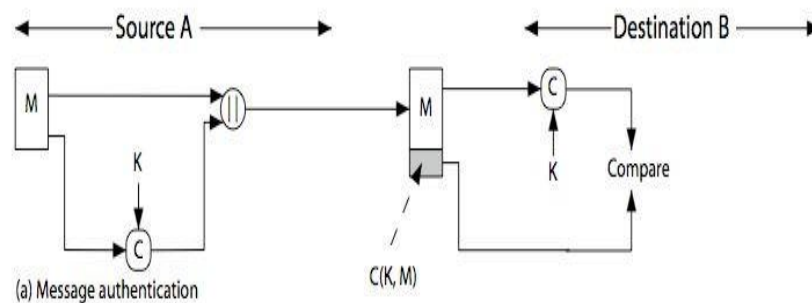
3. Basic terminologies definition
 - a. plaintext - original message
 - b. cipher text - coded message
 - c. cipher - algorithm for transforming plaintext to cipher text
 - d. key - info used in cipher known only to sender/receiver
 - e. Encryption - converting plaintext to cipher text
 - f. Decryption - recovering cipher text from plaintext

4. Cryptographic systems are characterized by

- a. type of encryption operations used
 - i. substitution / transposition / product
- b. number of keys used
 - i. single-key or private / two-key or public
- c. way in which plaintext is processed
 - i. block / stream

5. Message authentication code MAC works as follows

- ✓ Generated by an algorithm that creates a small fixed-sized block
- ✓ depending on both message and some key
- ✓ like encryption though need not be reversible
- ✓ appended to message as a signature
- ✓ receiver performs same computation on message and checks it matches the MAC
- ✓ provides assurance that message is unaltered and comes from sender



des

Where

M= input message

C = MAC function

K = shared secret key

MAC = message authentication code

6.

Conventional encryption

- The same algorithm with the same key is used for encryption and decryption.
- The sender and receiver must share the algorithm and the key
- One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.
- The sender and receiver must each have one of the matched pair of keys (not the same one).

Public key encryption

- One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.
- The sender and receiver must each have one of the matched pair of keys (not the same one)
- One of the two keys must be kept secret.

7. RSA encryption and decryption algorithms are stated as follows

| Key Generation | |
|--------------------------------------|---|
| Select p, q | p and q both prime, $p \neq q$ |
| Calculate $n = p \times q$ | |
| Calculate $\phi(n) = (p - 1)(q - 1)$ | |
| Select integer e | $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$ |
| Calculate d | $d \equiv e^{-1} \pmod{\phi(n)}$ |
| Public key | $PU = \{e, n\}$ |
| Private key | $PR = \{d, n\}$ |

8. The following describes

| Encryption | |
|-------------|-------------------|
| Plaintext: | $M < n$ |
| Ciphertext: | $C = M^e \bmod n$ |

| Decryption | |
|-------------|-------------------|
| Ciphertext: | C |
| Plaintext: | $M = C^d \bmod n$ |

IPSec services

- ✓ Access control
- ✓ Connectionless integrity
- ✓ Data origin authentication
- ✓ Rejection of replayed packets
- ✓ a form of partial sequence integrity
- ✓ Confidentiality (encryption)
- ✓ Limited traffic flow confidentiality

9. PGP operations for email authentication are

- a. sender creates message
- b. use SHA-1 to generate 160-bit hash of message
- c. signed hash with RSA using sender's private key, and is attached to message
- d. receiver uses RSA with sender's public key to decrypt and recover hash code
receiver verifies received message using hash of it and compares with
decrypted hash code