Himalaya's

CRACK JECA

ENTRANCE EXAM BOOK FOR MCA ASPIRANTS

- C PROGRAMMING
- OBJECT ORIENTED PROGRAMMING USING C++
- UNIX
- DATA STRUCTURE
- INTRODUCTION OF COMPUTERS
- OPERATING SYSTEM
- COMPUTER NETWORK
- DATABASE MANAGEMENT SYSTEM
- SOFTWARE ENGINEERING
- MACHINE LEARNING

- Dr. Pabitra Pal
- Dr. Partha Chowdhuri
- Mr. Prabhash Kumar Singh

Himalaya Publishing House

ISO 9001:2015 CERTIFIED

CONTENTS

CHAPTER 1

C PROGRAMMING

SYLLABUS

Variables and Data Types, I/O Operations, Operators and Expressions, Control Flow Statements, Functions, Array, Pointers, String Handling, Structures and Unions, Files Handling, Pre-processor Directives, Command Line Arguments.

1.1 INTRODUCTION

C is a general-purpose, procedural programming language. It was developed at Bell Labs by Dennis Ritchie, between 1972 and 1973. During the 1980s, C gradually gained popularity. C has been standardized by the American National Standards Institute (ANSI) since 1989 and subsequently standardized by the International Organization for Standardization (ISO).

A procedural language is a type of programming language that consists of a series of well structured steps and procedures within its programming context to write a program. It contains a systematic order of statements, functions and commands to complete a program. That means, it consists of a set of instructions and divides these instructions into smaller parts known as functions for the computer to perform.

1.2 KEYWORD AND IDENTIFIER

The keywords are reserved words. They have some special meanings to the compiler. The C programming language has 32 such keywords. The keywords are given in the Table 1.1. In C, an identifier is a name given to a variable, function, etc. Identifiers must be unique in a program. They are used to identify a particular entity in a program. For example:

```
int roll_number;
char name[10]
```

Here, **roll_number** and **name** are identifiers. While declaring any identifier, we need to keep in mind that a valid identifier can have **letters, digits and underscores only**. The first letter of an identifier **should not be a digit**. A keyword cannot be used as an identifier.

Table 1.1: Keywords in C Programming

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

1.3 DATA TYPES AND VARIABLES

In C, data type of a variable determines the type and size of data associated with that variable. For example

```
int marks;
```

Here, **marks** is a variable of integer (int) type with size 2 bytes or 4 bytes depending on the platform used. The **size of** operator is used to know the actual size of any data type in bytes. '**int**', '**char**', '**float**', and '**double**' are the basic data types in C. The '**short**' and '**long**' are **quantifiers** that are used to alter the size of *char*, *int* or *double* data types. Whereas, '**signed**' and '**unsigned**' are **sign qualifiers** which are used to specify the signed nature of integer types.

1.4 INPUT OUTPUT

The **printf()** function is one of the main output functions in the C programming language. The function is used to send formatted output to the standard output screen.

```
#include<stdio.h>
int main(){
printf("My First Program");
return 0;
}
```

A C program must contain the **main()** function. The execution of the source code begins from the **main()** function. Here, **printf()** is a library function that formats the string inside the double quotes and sends the formatted string as an output to the screen. The **printf()** function is declared in **stdio.h** file. So, the program should add the **stdio.h file in the program** using **#include<stdio.h>** to successfully run. This is generally added at the beginning of the source code. That is why it is called **a header file**. The statement **return 0;** inside **the main()** function is the "*exit status*" of the program.

In C programming, **scanf()** is one of the commonly used functions to take input from the user. The **scanf()** function reads formatted input from the standard input such as keyboard.

```
#include <stdio.h>
int main()
{
 int roll;
 printf("Enter a roll number:");
 scanf("%d", &roll);
 printf("Roll Number=%d", roll);
 return 0;
}
```

In this **program, %d** is a format specifier which is used in the **scanf()** function to take the input **(roll)** from the keyboard as an integer (int). When an integer is entered through the keyboard, it is stored in the roll variable.

1.4.1 Format Specifier

To format different data types inside the **printf** or **scanf** function, different format specifiers are used. For example, if we want to print the value of an **integer** variable using the **printf** function, we have to use %d as the format specifier.

```
int i=10;
printf("The value of i = %d", i);
```

The format specifiers for other data types are given in the Table 1.2.

Note that, both **%d** and **%i** format specifiers can be used for an integer type. Here, **%i** specifies integer whereas **%d** specifies signed decimal integer. For **printf** function, both these format specifiers can be used interchangeably. But, in **scanf** function **%d** and **%i** behaves differently. In **scanf**, the **%d** always takes integer value as **signed decimal integer**. Whereas, the **%i** takes an integer value as signed **decimal, hexadecimal** or **octal type**. So, the scanf function can take octal or hexadecimal input from the keyboard using **%i** only.

Table 1.2: Format Specifiers for Different Data Types

Data Type	Format Specifier
int	%d, %i
char	%c
float	%f
double	%lf
short int	%hd
long int	%ld, %li

unsigned int	"%u
long long int	"%lld, "%li
unsigned long int	"%lu
unsigned long long int	"%llu
long double	"%Lf

1.5 OPERATORS AND EXPRESSIONS

The C programming language supports different types of operators. The operators are used to perform operations on values and variables. The type of operators and their descriptions are given in the table below. These operators join different variables, functions and constants together to form expressions. For an example, consider the expression **fun() + x * 10**. Here, **fun()** is a function which may returns an integer, + and * are operators, x is a variable and 10 is a constant. Table 1.3 shows a list of operators used in C language.

Table 1.3: Type of Operators

Operator Type	Example	Description
Arithmetic Operators	+, -, *, /, %, ++, --	perform common mathematical operations
Assignment Operators	=, +=, -=, *=, /=	assign a value to a variable
Relational Operators	==, <, >, >=, <=, !=	compare two values
Logical Operators	&&, ||, !	return true/false comparing variables or values
Bitwise Operators	&, |, ^, ~	perform bitwise operator between variables or values
Increment/Decrement Operator	++, --	increment or decrement the value of a variable
Misc Operators	&, ?:	address of a variable and conditional expression

1.6 STORAGE CLASSES

The storage class of a variable determines whether the variable is stored in memory or CPU registers. It also determines the scope and initial value of the variable. There are four storage classes in C: automatic, register, static and external.

(a) Automatic: All variables defined within a function or block by default (or with auto specification) belong to the automatic storage class. The automatic variables are local to the block in which they are defined, and get destroyed on exit from the block.

(b) Register: The register specifier declares a variable of the register storage class. Variables declared as register are local to the block in which they are defined, and get destroyed on exit from the block. Register variables are placed in CPU registers, not in memory.

(c) Static: The static specifier gives the declared variable static storage class. Static variables can be used within a function. Static variables are not visible outside their function. But, static variables maintain their values between calls.

(d) External: The extern specifier gives the declared variable external storage class. No storage is allocated to an extern variable. It is assumed that the variable has already been defined elsewhere in the program. When we use an extern specifier the variable cannot be initialized because an extern variable is only declared, not defined.

1.7 FUNCTION

A function is a block of statements that perform a particular task. Every C program has at least one function which is **main()** function. We can divide our code into separate functions. A function declaration tells the compiler about a function's name, parameters, and return type. A function definition provides the body of the function.

Every function has four parts: **Function Name, Function Parameter, Function Body** and **Return Type**.

```
//int: return type
//sum: function name
//int m, int n: function parameters
int sum(int n, int m){
// function body
return m + n;
}
```

1.7.1 Recursive Function

In the C programming language, a program allows calling a function inside the same function. The function that calls itself is known as a recursive function and the technique is known as **recursion**. Every recursive function has at least one **base case** and one **recursive case**. The condition which returns from the function is called a **base case** and the condition which calls the function again is called the **recursive case**. An example of a recursive function is given below.

```
int fact(int n)
{
if (n <= 1)
   return 1; // base case
else
   return n*fact(n-1); // recursive case
}
```

1.8 LOOPS

Loops are used to execute a block of code repeatedly according to a condition given in the loop. There are 3 loops: for loop, while loop and do-while loop.

1.8.1 For loop

The syntax of for loop is:

```
for (initialization; condition; update){
  // code block
}
```

The **initialization** statement is executed only once. The **condition** expression is evaluated every time before the loop is executed. The **update** expression is executed at the end of each iteration. The loop is repeatedly executed until the condition expression is false. For loop is generally used when the number of iterations is known.

1.8.2 While Loop

The syntax of a while loop is:

```
while (condition){
  // code block
}
```

The while loop evaluates the condition before each iteration. If the condition is true, i.e., non-zero, codes inside the body of the while loop are executed. When the condition is false or zero, the while loop is terminated.

1.8.3 Do-while Loop

The syntax of a do-while loop is:

```
do{
  // code block
}while(condition);
```

The code inside the loop is executed once. Then, the condition is evaluated. If the condition is true or non-zero, the code inside the loop is executed again. When the condition is false or zero, the do-while loop is terminated.

The do-while loop is similar to the while loop. But there is an important difference. The body of the do-while loop is executed once before checking the condition. So, unlike while loop, a do-while loop is executed at least once.

1.8.4 Break Statement

The break statement terminates the loop immediately when it is encountered. The break statement is almost always used with some conditional statement inside the loop.

The following loop terminates whenever the number 5 is encountered.

```
# include <stdio.h>
int main(){
   int number;
   for(number=1; number <= 10; number++) {
      printf("%d", number);
      // If the number is 5, the loop terminates
      if(number == 5){
         break;
      }
   }
}
```

1.8.5 Continue Statement

The continue statement skips all the statements declared after it inside the loop and continue with the next iteration. The following loop skip printf statement whenever the number 5 is encountered.

```
#include<stdio.h>
int main(){
   int number;
   for(number=1; number <= 10; number++) {
      // If the number is 5, the loop continue with the next iteration.
      // without executing the next printf statement
      if(number == 5){
         continue;
      }
      printf("%d", number);
   }
}
```

1.9 ARRAY

1.9.1 What is an Array?

An array is a collection of a **fixed number** of items of the **same type**. The items are stored **sequentially** in the memory. Arrays are of two types:

- **One-dimensional arrays and**
- **Multi-dimensional arrays**

1.9.2 Declare and Initialize an One-dimensional Array

To store 10 integers in sequence, we can create an array like this:

```
int roll[10];
```

An array can be initialized as:

```
int marks[5] = {2, 4, 6, 8, 10};
```

1.9.3 Access the Elements of One-dimensional Array

Array elements are accessed by indices. Array index starts from 0. So, to access the 3rd element of the **marks** array, we can write:

```
int n = marks[2]; // here, [2] indicates the third element of the array
```

Similarly, to modify the 2nd element of the array, we can write:

```
marks[1] = 1; // here, [1] indicates the second element of the array
```

1.9.4 Important Properties of Array

(a) Array index always starts with 0, not 1.

(b) An array element can be accessed by indices.

(c) An array will always hold items of the same type.

(d) The size and type of an array are fixed. Once declared, the size and type of the array cannot be changed.

1.9.5 Declare and Initialize a Two-dimensional Array

To store 6 integers in a two-dimensional array or 2-D array with 2 rows and 3 columns, we can create a 2-D array like this:

```
int cell[2][3];
```

A 2-D array can be initialized as:

```
int cell[2][3] = {
                {2, 4, 6},
                {8, 10, 12}
                };
```


For any 2-D array, at least the column index should be specified at the time of initialization. So, the following declaration is correct.

```
int cell[ ][3] = {1, 2, 3};
```

But, the following declaration is not correct.

```
int cell[ ][ ] = {1, 2, 3};
```

1.10 STRING

In C programming, a string is an array of characters terminated with a null character ('0').

1.10.1 Declaring a String

This is how we can declare a string:

```
char student[30];
```

Here, the **char** is the type of the array, **student** is the name and 30 is the size of the array. It means, the student array can store up to 30 characters, including '0' the null character, in contiguous memory locations.

1.10.2 Initialize a String

There are many ways to initialize a string in C:

```
// the size of the "Hello World" string becomes the size of the array here
char name[] = "Hello World";
OR char name[] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '\0'};
// the size of the array is 30 here
char name[30] = "Hello World"; OR
char name[30] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '\0'};
```

As strings are also an array, a particular element or character can be accessed using an array index.

For example, to access the fifth element of the name array, we can write:

```
char e = name[4]; // here, [4] indicates the 5th character of the 'name' array
```

Similarly, to change the 10th character of the name array, we can write:

```
name[9] = 'R'; // here, [9] indicates the 10th character of the 'name' array
```

1.10.3 Commonly Used String Handling Functions

Some of the commonly used library functions for string handling are as follows:

- strlen
- strcat
- strcmp
- strcpy

The **strlen()** function calculates the length of a string. The **strcpy()** function copies a string to the another string. The **strcmp()** function compares two strings. The **strcat()** function concatenates two strings.

```
#include <stdio.h>
#include <string.h>
int main(){
char str1[25] = "Vidyasagar";
char str2[25] = "University";
int n;
// calculates the length of str1 string and store in variable n.
n = strlen(str1);
printf("%d\n", n);
   n = strcmp(str1, str2);
if(n==0){
   printf("Strings are same\n");
}else{
   printf("Strings are not same\n");
}
   //concatenates str1 and str2 and the resultant string is stored in str1
   strcat(str1, str2);
   printf("%s\n", str1); // prints "Vidyasagar University"

   // copies the content of str2 to str1
   strcpy(str2, str1);
   printf("%s\n", str2); // This output will be "Vidyasagar University"

   return 0;
}
```


1.11 POINTER

A pointer is a variable that can store the address of another variable. Every variable resides in a memory location and every memory location has its own address. The address of a variable can be accessed by an ampersand (&) operator. Every pointer has a type and the pointer can store the address of another variable of the same type.

1.11.1 Define a Pointer Variable

A pointer can be defined and initialized as follows:

```
int *ip;        // definition: ip is a pointer to an integer
int i=10;       // i is an integer
ip = &i;        // initialization: address of i is assigned to pointer ip;
float *fp;      // definition: fp is a pointer to a float float f=10.5; // f is a float
fp = &f;        // initialization: address of f is assigned to pointer fp
```

1.11.2 Using Pointers in Function

The following program makes use of pointers to swap the values of two variables.

```
#include <stdio.h>
void swap(int *x, int *y){
   int temp;
   temp=*x;
   *x = *y;
   *y = temp;
}
int main(){
   int a=5;
   int b=6;
   printf("a = %d, b = %d", a, b); // prints a = 5, b = 6
   swap(a, b);
   printf("a = %d, b = %d", a, b); // prints a = 6, b = 5
}
```

1.12 STRUCTURE AND UNION

Sometimes, it is required to store some related elements under the same name. For example, a student details like roll, name and marks need to be stored together. The C language provides structures to handle this type of situation. A structure can be defined as a single entity that can

store some related variables with different data types. Using a structure variable, it is possible to access all the data members inside a structure.

Like structures, unions are another type of user-defined data type. They are used to store a collection of variables with different data types. But unlike structure, the unions can store information of any one type of variable at a time.

1.12.1 Syntax to Define a Structure and Union

```
struct NAME1{
   // structure definition
   datatype1 variable1;
   datatype2 variable2;
   datatype3 variable1;
   ---
};
union NAME2{
   // structure definition
   datatype1 variable1;
   datatype2 variable2;
   datatype3 variable1;
   ---
};
```

Here, 'struct' is the keyword to define a structure. 'NAME' is the structure name. 'datatype' indicates the actual data type of the variable in the structure. Similarly, '**union**' keyword is used to declare a union.

1.12.2 Declaring Structure and Union Variable

The structure or union variables can either be defined at the time of definition or separately using the structure or union name followed by variable names. A 'student' type structure and a 'course' type union are given below.

```
struct student{
   // defining the student structure
   int roll;
   char name[20];
   int marks;
}student1, student2; // declaring student1 and student2 (one way)
struct student student3, student4; // declaring student3 and student4 (another way)
```



```
union course{
   // defining the student structure
   char subject_name[20];
   char teacher_name[20];
   int course_id;
}course1; // declaring course1 (one way)
union course course2; // declaring course2 (another way)
```

1.12.3 Initializing and Using Structure Variables

Let's see an example to understand how a structure variable is initialized and used in a program.

```
#include<stdio.h>
#include <string.h>
struct student
{
   int roll;
   char name[50];
   int marks;
}student1; //declaring variable for structure
int main( ){
   // declare another student variable here
   struct student student2;
      //store first student information
   student1.roll = 1;
   strcpy(student1.name, "Mark");
   student1.marks = 87;
      //store second student information
   student2.roll = 2;
   strcpy(student2.name, "Bill");
   student2.marks = 86;
   //printing first Student1 information
   printf("Roll = %d\n", student1.roll);
   printf("Name = %s\n", student1.name);
   printf("Marks= %d\n", student1.marks);
```

```
    //printing second Student2 information
    printf("Roll = %d\n", student2.roll);
    printf("Name = %s\n", student2.name);
    printf("Marks= %d\n", student2.marks);
        return 0;
}
```

1.12.4 Size of Structure and Union

The total size of a structure is the sum of sizes of all variables declared in the structure. Whereas, the total size of a union is equal to the size of the variable with the maximum size. This is because a union can store information of any one type of variable at a time. Let's see an example to understand how the size of struct and union differs.

```
struct student{
    int roll;
    char name[20];
    int marks;
};
union course{
    char subject_name[20];
    char teachar_name[30];
    int course_id;
};
```

Assuming the size of int is 4 bytes and size of character is 1 byte, the size of the structure 'student' will be (4 + 20 + 4) = 28 bytes. Whereas, the size of the union 'course' will be 30 which is the maximum size of a variable in the union.

1.13 COMMAND-LINE ARGUMENT

In the C programming language, it is possible to pass parameter values to the main function before the execution of any program. As the arguments are passed through the command line, these arguments are called command-line arguments.

In general, we write the **main()** function in our program without any parameters. But, to implement command-line argument in a program we need to write **main()** function with **two parameters**.

```
int main (int argc, char *argv[])
```


In the state mentioned above, the first argument is **argc**, which is actually the number of arguments passed to the main function. The second argument, **argv[]** is an array of pointers to command line parameter strings. The **argv[0]** pointer points to the executable file name or the program name. Similarly, argv[1] pointer points to the first parameter passed as a string to the main function and so on.

```
#include <stdio.h>
int main( int argc, char *argv[]){
    // 'argv[0]' is the first parameter, which is executable file name
    printf("Name of the program = %s\n", argv[0]);
    if( argc > 1 ) {
        // first command line argument
        printf("The first command line argument = %s\n", argv[1]);
    }
    if( argc > 2 ){
        // second command line argument
        printf("The second command line argument = %s\n", argv[2]);
    }
    return 0;
}
```

The following program displays all the command-line arguments utilizing 'argc' which gives the number of arguments.

```
#include <stdio.h>
int main( int argc, char *argv[]){
    int i=0;
    printf("Number of argument = %d\n", argc);
    printf("Name of the program = %s\n", argv[0]);
    if( argc > 1 ) {
        for(i=1; i<argc; i++){
            printf("Argument Number %d = %s\n", i, argv[i]);
    }
    }
    return 0;
}
```

1.14 PRE-PROCESSOR DIRECTIVES

The pre-processor is a program that modifies the source code before the compiler compiles the code. Pre-processor directives are actually the instructions that are used to identify a code that is to be pre-processed before the actual compilation. Every pre-processor directive is prefixed by the symbol #. Some of the pre-processor directives are given in the Table 1.4 along with their interpretation.

Table 1.4: Different Pre-Processor Directives

Directive	Interpretation
#include	includes a file to the source code
#define	defines a macro
#undif	undefines a macro
#ifdef	tests if a macro is defined
#ifndef	tests if a macro is not defines

1.15 TYPEDEF

A variable in C can be given a meaningful name using the keyword 'typedef'. We can think of it as an alias for a variable. But, why do we use 'typedef' to give a variable another name? This keyword is actually useful when we are dealing with the long data type like structure declarations or pointer declarations.

Syntax: typedef <name> <another_name>;

Let's understand the concept through some simple declarations.

```
//typedef with variable
typedef unsigned long int ulint;
ulint i=9999;
//typedef with structure
typedef struct student stud;
stud student1, student2;
//typedef with pointer
typedef void (*f)(int) fnptr;
fnptr f1;
//typedef with array
typedef int myarray[10];
myarray x;
```

The #define directive and typedef may seem similar in C programming. But they have some differences. The 'typedef' operator can give symbolic names to types only. It is interpreted by the

compiler. Scope rule is followed as usual in case of 'typedef'. Whereas, '#define' can define an alias for symbolic names as well as values and no scope rule is followed. It is interpreted by the pre-processor.

1.16 FILE HANDLING

The C programming language supports I/O operations on files stored in the secondary storage devices. It can handle both text files and binary files. The operations that can be performed on a file are given below.

- Creating a new file
- Opening an existing file
- Reading from a file
- Writing to a file
- Closing a file

Files can be opened with different modes for different types of operations. The modes are **r, w, a, r+, w+** and **a+**. **r** mode is used for reading a file only. **w** and **a** modes are for writing and appending to a file, respectively. If no file exists, a new file is created. The **r+, w+** and **a+** modes are used for reading and writing to a file. In the case of w+, if no file exits, a new file is created and if the file exits it gets blank before writing. Whereas, in a+ mode, the content is appended to the existing file. The **fopen** function is used to open a file. The first parameter is the file path and the second parameter is the opening mode (r, w, a, r+, w+, a+). It returns a FILE pointer. **FILE** is actually an in-built structure that holds different information about a file. The **fclose** function is used to close a file. It takes FILE pointer as the parameter. A sample program is given below for a better understanding of file handling.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
   // declare a name array char name[30];
   // declare a FILE pointer
   FILE *fptr;
   // open myfile.txt file with 'w' or write mode
   fptr = fopen("C:\\myfile.txt","w");
   // enter a number
   printf("Enter a name:");
   scanf("%s",name);
   // write the name to the file using file pointer
   fprintf(fptr,"%s",name);
```

```
// close the file
fclose(fptr);
// open the file again for reading in read or 'r' mode
fptr = fopen("C:\\myfile.txt","r");
// read the name from the file using file pointer
fscanf(fptr,"%s", name);
// display the name in standard output
printf("The name is %s", name);
// close the file
fclose(fptr);
return 0;
}
```

1.17 SOLVED QUESTIONS

1. Which is valid C expression?
   A. int my_num = 100,000;
   B. int my_num = 100000;
   C. int my num = 1000;
   D. int 1my_num = 10000;

   Solution: (B). A variable name *cannot contain a space*, a number cannot contain any character except digits only, and a variable *cannot start with a digit*.

2.
```
struct a
{
  int a;
  char b;
  int c;
};
Union b
{
  char a;
  int b;
  int c;
};
```
Which of these is correct?

   A. Size of a is always different from size of b.
   B. Size of a is always same as size of b.
   C. a and b cannot be same because of non-homogeneity
   D. It might result in runtime error.

   Solution: (A). The size of a struct is the sum of the sizes of its variables. Whereas, the size of an union is equal to the size of the variable having the maximum size. So, size of a struct is different from size of a union.

3. Which of the following cannot be a variable name in C?
   A. volatile
   B. true
   C. friend
   D. export

   Solution: (A). *volatile* is a keyword. So, it cannot be a variable name.

4. Which keyword is used to prevent any changes in the variable within a C program?
   A. immutable
   B. mutable
   C. const
   D. volatile

   Solution: (C). *const* keyword makes any variable immutable or unchangeable.

5. What will be printed after the execution of the following program code?

```
main()  {  printf("\nab");  printf("\bcd");
printf("\ref");
}
```

   A. abd
   B. efd
   C. bcd
   D. abedef

   Solution: (B). The | sign represents the position of the cursor. The problem is discussed step by step here:

```
printf("\nab");-
ab|
```
[\n=newline, prints 'ab' in the newline]
```
printf("\bcd");a|
```
[\b=backspace, deletes b and put the cursor after 'a']
```
acd| [then inserts 'cd']
printf("\ref");-
lncd
```
[\r=carriage return, put the cursor at the beginning of the line]
```
efd
```
[and prints 'ef' replacing 'ac' and put the cursor after 'ef']
So, the ultimate output will be 'efd'

6. sizeof(int) is _____.
   A. always 2 bytes.
   B. depends on compiler being used.
   C. Depends on computer hardware being used.
   D. always 32 bits.

   Solution:(B). The size of operator returns the size of a variable or a data type in bytes. All fundamental data types and their sizes are compiler-dependent and only compiler-dependent.

7. typedef char *ch; const ch P; In this code, what is P?
   A. P is a constant.
   B. P is a character constant.
   C. P is a character type.
   D. P is a pointer of character type.

   Solution: (A). The line 'const ch P' will be converted to 'const char* P'. Therefore, P is a const character pointer here.

8. Which of the following is true for variable names in C?
   A. They can contain alphanumeric characters as well as special characters.
   B. It is not an error to declare a variable to be one of the keywords (like goto, static).

C. Variable names cannot start with a digit

D. Variable can be of any length

Solution: (C). A variable name can use any alphabet, digit or underscore (_) character. The name cannot start with a digit.

9. Which of the following is NOT possible with any 2 operators in C?

A. Different precedence, same associativity

B. Different precedence, different associativity

C. Same precedence, different associativity

D. All of the mentioned

Solution: (C). The operators with *same precedence* will always have *same associativity*.

10. What is #include <stdio.h>?

A. Pre-processor directive

B. Inclusion directive

C. File inclusion directive

D. None of the mentioned

Solution: (A). Every valid C statement starts with a # sign is called a *pre-processor directive*.

11. Which of the following is not possible statically in C language?

A. Jagged Array

B. Rectangular Array

C. Cuboidal Array

D. Multidimensional Array

Solution: (A). A *jagged array* is an two-dimensional array in which every row may have different number of columns. In C, jagged array cannot be declared.

12. A program to find the factorial of a given number is written using recursion and without using recursion. Which of these will result in stack overflow for a given number?

A. Only first program with recursion

B. Both may result for the same number

C. Only second program

D. None of these

Solution: (A). A *recursive function* will always use a stack for storing some data like local variables, arguments and return address for every function call. If the number of recursive calls becomes very high, there may not be enough space in the stack to store these data. In that case, *stack overflow* can occur.

13. In C language, FILE is of which data type?

A. int

B. char *

C. struct

D. None of the mentioned

Solution: (C). The *FILE is a struct* in C. When a file in C is opened it always returns a pointer of type *FILE*.

14. Consider the following C declaration:

```
struct {
short s[5]
 union{
float y; long z;
}u;
}t;
```


Assume that objects of the type short, float and long occupy 2 bytes, 4 bytes and 8 bytes, respectively. The memory requirement for variable t, ignoring alignment considerations, is

A. 22 bytes

B. 18 bytes

C. 14 bytes

D. 10 bytes

Solution: (B). The size of struct t will be equals to *size of s[5] plus size of z* in union u. Because, the size of an union is equals to the size of the variable with maximum size. So, the total size will be (5*2) + (8) = 18 bytes.

15. What is the meaning of the following C statement? printf("%10s", state);

A. 10 spaces before the string state is printed

B. Print empty spaces if the string state is less than 10 characters

C. Print the last 10 characters of the string

D. None of the mentioned

Solution: (B). The "%10s" option prints *empty spaces to the left* of a string if the string is not 10 characters long. In the same way "%10s" option prints empty spaces to the right of a string if the string is not 10 characters long.

16. Variables of function call are allocated in _____.

A. Registers and stack.

B. Cache and heap.

C. Stack and heap.

D. Registers and heap.

Solution: (C). With a function call a statically allocated variable is stored in the stack. Whereas, a dynamically allocated variable is stored in the *heap*.

17. What does this declaration mean?

int X : 4;

A. X is a four-digit integer.

B. X cannot be greater than a four-digit integer.

C. X is a four-bit integer.

D. None of the these

Solution: (C). The : operator is called a *bit field operator*. It specifies the *number of bits* that will be allocated for the variable.

18. Why is a macro used in place of a function?

A. It reduces execution time.

B. It reduces code size.

C. It increases execution time.

D. It increases code size.

Solution: (A). The main benefit of using a macro is faster execution time. During pre-processing, a macro is replaced by its definition each time it is used. In case of a function, some time is wasted for *call* and *return*.

19. Which one of the following is a loop construct that will always be executed once?

A. for

B. while

C. switch

D. do while

Solution: (D). In a do-while loop the condition is written *at the end of the loop*. So, it is executed at least once before checking the condition.

20. Determine the output of the C code mentioned below:

```
#include <stdio.h>
int main(){
    float q = 'a';
    printf("%f", q);
    return 0;
}
```

A. run time error

B. a

C. 97.000000

D. 0000000

Solution: (C). The char 'a' returns the ascii value of 'a' which is 97 to the variable q. When printed, the q prints 97 as a float value as 97.000000.

21. Which one of the following is equivalent to the expression str[4]?

A. str + 4

B. *str + 4

C. *(str + 4)

D. str[3]

Solution: (C). An array *str[n]* can be written as **(str + n) or as *(n + str)*.

22. Out of the following, which one is not valid as an if-else statement?

A. if ((char) x)

B. if (x)

C. if (func1 (x))

D. if (if (x = 1))

Solution: (D). The condition of ' if ' statement should always be any value, either positive or negative. The condition is false for 0 and true for any *non-zero* value.

23. We cannot use the keyword 'break' simply within _____.

A. while

B. for

C. if-else

D. do-while

Solution: (C). The 'break' keyword can only be used within a loop.

24. Out of the following operations, which one is not possible in the case of a register variable?

A. Global declaration of the register variable

B. Copying the value from the memory variable

C. Reading any value into the register variable

D. All of the above

Solution: (A). The 'register ' variable can only be declared as a local variable.

25. Comment on the C statement given below:

int (*p)[5];

A. A ragged array

B. An array 'p' of pointers

C. A pointer 'p' to an array

D. None of the above

Solution: (C). *int (*p)[5]* : p is a pointer to an array of 5 integers. *Int *p[5]* : p is an array of 5 integer pointers.

Answers:

1. B	2. A	3. A	4. C	5. B
6. B	7. A	8. C	9. C	10. A
11. A	12. A	13. C	14. B	15. B
16. C	17. C	18. A	19. D	20. C
21. C	22. D	23. C	24. A	25. C

1.18 PRACTICE QUESTION SETS

1. If you do not specify a storage class for a variable, _____.

A. You get compiler error.

B. You get a compiler warning.

C. Output is null always

D. None of the above

2. Which of these properties of #define is false?

A. These always obey the scope rules

B. We can make use of a pointer to #define

C. The #define can be externally available

D. All of the above

3. The correct format of declaring a function is _____.

A. type_of_return name_of_function (argument type);

B. type_of_return name_of_function (argument type)

C. type_of_return (argument type) name_of_function;

D. All of the above

4. We can test the presence of a loop in a linked list by _____.

A. Comparing the node's address with the address of all the other nodes

B. Travelling the list. In case we encounter the NULL, then no loop exists

C. Comparing the stored values of a node with the values present in all the other nodes

D. None of the above

5. Determine what's true for x, if we define x as "int *x[10];"?

A. This definition will only allocate 10 pointers but will not initialize them

B. The initialization must be explicitly performed

C. The definition will only allocate 10 pointers but will not initialize them. Also, the initialization has to be explicitly performed

D. Error

6. Which of these is the correct initialization method for the following:

typedef char *string;

A. More than a single space shouldn't be given when we are using typedef

B. *string p = 'A';

C. string p = "Hello";

D. *string *p = "Hello";

7. We can determine the size of a union with the help of the size of

A. The sum of all the member's sizes

B. The biggest member of the union

C. The last member of the union

D. The first member of the union

8. Out of the following snippet, which one will generate random numbers effectively?

A. rand(time(NULL));

B. rand(10);

C. rand();

D. All of the above

9. We can achieve the modulus for float by _____.

A. x*e y
B. modulus(x, y);
C. fmod(x, y);
D. mod(x, y);

10. The definition of the function abort() is in which header file?
A. stdlib.h
B. assert.h
C. stdio.h
D. stdarg.h

11. What is the primary purpose of the pre-processor directive #error?
A. Rectifies the first error occurring in the code
B. Rectifies the errors present in a code
C. Causes a preprocessor to ignore any error
D. Causes a pre-processor to report some fatal error

12. _____ is a condition in which the memory is dynamically reserved but isn't accessible to any program.
A. Pointer Leak
B. Frozen Memory
C. Dangling Pointer
D. Memory Leak

13. _____ tells a compiler that the data would be defined somewhere, and it would be connected to the linker
A. variable
B. yvals
C. errno
D. extern

14. All keywords in C are in _____
A. LowerCase letters
B. UpperCase letters
C. CamelCase letters
D. None of the mentioned

15. Which of the following cannot be a variable name in C?
A. Volatile
B. True
C. Friend
D. Export

16. What is 'short int' in C programming?
A. The basic data type of C
B. Qualifier
C. Short is the qualifier and int is the basic data type
D. All of the mentioned

17. Which keyword is used to prevent any changes in the variable within a C program?
A. immutable
B. mutable
C. const
D. volatile

18. Choose a correct statement about C language arrays.
A. An array address is the address of first element of array itself.
B. An array size must be declared if not initialized immediately.
C. Array size is the sum of sizes of all elements of the array.
D. All the above

19. What is the result of logical or relational expression in C?

A. True or False
B. 0 or 1
C. 0 if an expression is false and any positive number if an expression is true
D. None of the mentioned

20. An array Index starts with?
A. −1
B. 0
C. 1
D. 2

21. Choose correct statement about Functions in C Language.
A. A Function is a group of c statements which can be reused any number of times.
B. Every Function has a return type.
C. Every Function may not return a value.
D. All of the above

22. What is the output of C Program?

```
int main() {
  int a[];
  a[4] = {1,2,3,4};
  printf("%d", a[0]);
  return 0;
}
```

A. 1
B. 2
C. 4
D. Compiler error

23. An entire array is always passed by _____ to a called function.
A. Call by value
B. Call by reference

C. Address relocation
D. Address restructure

24. Which of the following typecasting is accepted by C language?
A. Widening conversions
B. Narrowing conversions
C. Widening and Narrowing conversions
D. None of the mentioned

25. Where in C the order of precedence of operators do not exist?
A. Within conditional statements, if, else
B. Within while, do-while
C. Within a macro definition
D. None of the mentioned

26. char a[5]="hello"
Which of the following is correct?
A. In array we cannot do the operation
B. Size of a is too small
C. Size of a is too large
D. Runtime error

27. Choose correct statement about C array pointers.
A. You can compare two array elements with *p = *(p+i)
B. You can compare two pointers with p1= p2.
C. Accessing out of bounds index element is valid and it returns a garbage value.
D. All of the above

28. Choose a correct statement about C structures.

A. Structure elements can be initialized at the time of declaration.

B. Structure members can not be initialized at the time of declaration

C. Only integer members of structure can be initialized at the time of declaration

D. None of the above

29. Global variables in different files are checked

A. at load time.

B. at assembly time.

C. at linking time.

D. during execution time.

30. Indirect addressing uses:

A. Arrays

B. Pointers

C. Structures

D. Array with pointers

31. How many values can a C Function return at a time?

A. Only one value

B. Maximum of two values

C. Maximum of three values

D. Maximum of 8 values

32. Consider the following C function definition:

```
int Trial (int a, int b, int c){
  if((a>=b)&& (c <b)
    return b;
  else if (a >=b)
    return Trial (a, c, b);
  else return Trial (b, a, c);
}
```


The function Trial _____.

A. Finds the maximum of a, b and c

B. Finds the minimum of a, b and c

C. Finds the middle number of a, b and c

D. None of these

33. What is the output of C Program with arrays?

```
int main()
{
  int ary[][2][3] = {
    {{1,2,3},{4,5,6}},
    {{7,8,9},{10,11,12}}
  };
  printf("%d%d",ary[0][0][0],
    ary[1][1][1]);
  return 0;
}
```

A. 1 12

B. 1 11

C. 7 12

D. 1 6

34. Which of the following are C pr processors?

A. #ifdef

B. #define

C. #endif

D. All of the mentioned

35. What are types of Functions in Language?

A. Library Functions

B. User-defined Functions

C. Both Library and User-defined

D. None of the above

36. Choose correct statements about C Language Pass By Value.

A. Pass By Value copies the variable value in one more memory location

B. Pass By Value does not use Pointers

C. Pass By Value protects the original variables from being changed in outside functions

D. All of the above

37. What is the first step in C program building process?

A. Compiling

B. Assembling

C. Linking

D. Pre-processing

38. What is a Pragma in C language?

A. A Pragma may be an instruction to build tool to process or generate comments

B. A Pragma may be an instruction to compiler to execute specific functions at specific times say startup or exit of program

C. A pragma may be an instruction to tell compiler to ignore certain warnings

D. All of the above

39. What is an array in C language?

A. A group of elements of same data type.

B. An array may contain more than one element

C. Array elements are stored in contiguous memory locations

D. All of the above

40. Which of these is NOT a relational or logical operator?

A. =

B. ||

C. ==

D. !=

41.

```
#define f(a,b) a+b
#define g(a,b)a*b
main() {
int m; m=2*f(3,g(4,5));
printf("n m is %d", m);
}
```

What is the value of m?

A. 70

B. 50

C. 46

D. 69

42. The global variables are _____.

A. External

B. Internal

C. Both External and Internal

D. None of the above

43. Choose correct syntax for C arithmetic compound assignment operators.

A. a+=b is (a=a+b)

B. a-=b is (a=a-b)

C. a*=b is (a=a*b)

D. All of the above

44. Choose a correct C statement about number 66 in the array:

```
int ary[3][2] =
{{11,22},{33,44},{55,66}};
```

38

A. ary[2][1]
B. *(*(ary+2)+1)
C. *ary[2]+1
D. All of the above

45. What is the way to suddenly come out of or quit any loop in C Language?
A. continue statement
B. break statement
C. leave statement
D. quit statement

46. What is the output of C Program?

```
int main()
{
while(true) {
printf("Hello");
break;
}
return 0;
}
```

A. Hello
B. Hello is printed unlimited number of times
C. No output
D. Compiler error

47. What is a structure in C language?
A. A structure is a collection of elements that can be of same data type.
B. A structure is a collection of elements that can be of different data type.
C. Elements of a structure are called members.
D. All of the above

48. What is the size of a C structure?
A. C structure is always 128 bytes.
B. Size of C structure is the total bytes of all elements of structure.
C. Size of C structure is the size of largest element.
D. None of the above

49. Leftmost bit 0 in singed representation indicates _____.
A. A Positive number
B. A Negative Number
C. An Unsigned number
D. None of the above

50. A function which calls itself is called a _____ function.
A. Self Function
B. Auto Function
C. Recursive Function
D. Static Function

Answers:

1. D	2. D	3. A	4. A	5. C
6. C	7. B	8. C	9. C	10. A
11. D	12. D	13. D	14. A	15. A
16. C	17. C	18. D	19. B	20. B
21. D	22. D	23. B	24. C	25. D
26. B	27. D	28. B	29. C	30. B
31. A	32. D	33. B	34. D	35. C
36. D	37. D	38. D	39. D	40. A
41. C	42. A	43. D	44. D	45. B
46. A	47. D	48. B	49. A	50. C

❐ ❐ ❐

CHAPTER 2

OBJECT ORIENTED PROGRAMMING USING C++

SYLLABUS

Data Types, If/Else, Loops, Function, Switch Case, Pointer, Structure, Array, String, Function Overloading, Function Templates, SCOPE of Variable, Type Aliases (typedef/using), Unions, Enumerated Types (enum), Class, Constructors, Overloading Constructors, Member Initialization in Constructors, Pointers to Classes, Overloading Operators, Keyword 'this', Static Members, Const Member Functions, Class Templates, Template Specialization, Namespace, Friendship (Friend Functions and Friend Classes), Inheritance, Polymorphism, Virtual Members, Abstract Base Class.

2.1 INTRODUCTION

In procedure-oriented programming, the instructions are organized into groups known as functions. In a multi-function program the function may have its own local data or the functions can access any data outside the function. These data are called global data as they can be accessed by any function globally. When the program becomes very large it makes it difficult to find out what data or data structure is used by which function. If the external data structure needs to be changed, all other functions that access the data structure also need to be changed. This introduces unnecessary bugs in the program. Moreover, as procedure-oriented programming is action-oriented, it cannot be used to design any suitable model for real-world problems.

Object-oriented programming (OOP) was invented to overcome the drawbacks of procedure-oriented programming. In OOP, the data is accessible throughout the program. The data is strongly coupled with the function that operates on it. It protects the data from any un-intentional modification from outside the function. The OOP decomposes a problem into a number of classes and the classes encapsulate the data as well as the functions that operate on it. An instance of a class is called an object. Any data inside an object can only be accessed by the function associated with that object.

2.2 DATA TYPES

The variables in a programming language are used to store data values. In C++, there are different types of variables for storing different types of data. The data types and the type of values they can store are given in the following table:

Data Type	Size	Description
int	4 bytes	stores integers
float	4 bytes	stores fractional numbers up to 4 byte
double	8 bytes	stores fractional numbers up to 8 byte
bool	1 byte	stores true or false values
char	1 byte	stores a single character

2.3 INPUT AND OUTPUT

The ***cin*** and ***cout*** are the standard in-built objects in C++ for *reading* and *writing* value of a variable. The ***cout*** object, along with operator ≪, is used to display/print strings and value of a variable. Whereas, the ***cin*** object, along with operator ≫ is used to take input from the keyboard. The program below shows how to get input of a variable from the keyboard and how to display a string and value of a variable.

```
#include <iostream>
using namespace std;
int main() {
   int i;
   cout << "Enter the value of i";
   cin >> i;
   cout << "The value of i is "<< i ;
   return 0;
}
```

2.4 SCOPE OF VARIABLES

Based on their scopes, a variable can be classified as: **Local variables and Global variable**. A variable declared inside a block or a function is called a **local variable**. Local variables can only be used inside the function or the block in which they are declared. A block in C++ starts with "{" and ends with "}".

The **global variables** are declared outside all the functions and blocks. So, any function or any block can access the global variables. The global variables are generally declared at the beginning of any source code. The code given below describes how a variable can be accessed depending on its scope.


```
#include<iostream>
using namespace std;
int g=100;
int main(){
  //scope of x starts here
  int x=4;
  {
     //scope of y starts here int y=5;
     {
        // scope of z starts here
        int z=6;
        cout<<g<<x<<y<<z<<endl; // OK: g is a global variable and
                                // x,y,z are in scope
     }// scope of z ends here
     cout<<x<<y<<z; // ERROR! z is not in this scope
     cout<<x<<y; // OK
  }//scope of y ends here
  cout<<x<<y; // ERROR! y is not in this scope
  cout<<x; // OK: x is in scope
  // scope of x ends here
  return 0;
}
```

2.5 CONDITIONAL STATEMENT

The **if** and switch-case statements are known as conditional statements as they execute a block of statements depending on a condition.

2.5.1 If

The **if** statement is used to specify a block of code to be executed if the condition in it becomes true. The **if** statement can also be used with **else** or **else-if** statements. The working principle of these statements can be well understood by the following example.

Example:

```
if(x > 60){
   // code block A
}else if(x > 40){
```

```
    // code block B
}else{
    // code block C
}
```

If the value of x is greater than 60, then code block A will be executed. If the x value is greater than 40, then the code block B is executed. Otherwise, code block c is executed.

2.5.2 Switch-case

The switch statement is used to select any one of many code blocks to be executed.

Syntax:

```
switch(expression) {
  case a:
    // any code here break;
  case b:
    // any code here break;
  default:
    // any code here
}
```

The switch-case statement works in the following manner:

- The expression in the **switch** statement is evaluated first
- The value of the expression is compared with the value of the **case** statements one by one
- If the value matches with any **case** statement, the code under that case statement is executed
- The **break** statement after every case block ensures that only one case block is executed for one **switch** statement.
- The '**default**' block is executed when **no match is found in the case statements**. It is an optional block.

2.6 LOOP

A loop is used to execute a block of statements as long as the condition specified in the loop remains true. In C++, three types of loops are available; *for* loop, *while* loop, and do-while loop.

2.6.1 For Loop

The for loop is used to loop through a block of code when we know exactly how many times you want.

Syntax:

```
for (Statement 1; Statement 2; Statement 3) {
  // code block
}
```

Here, ***Statement 1*** is executed first, before the execution of the code block. ***Statement 2*** is the condition that is checked every time before executing the code block. ***Statement 3*** is executed every time after the code block is executed.

The code below prints the numbers from 1 to 100.

```
for (int n = 1; n <= 100; n++) {
cout << n << endl; // endl is newline object in C++
}
```

2.6.2 While Loop

The **while** loop is generally used in a situation when we don't know exactly how many times **the block of code** will be executed. The while loop through the **block** of code as long as the loop *condition* is true.

Syntax:

```
while (condition) {
  // code block
}
```

Example: The code below prints the numbers from 1 to 100.

```
int n = 1;
while (n <= 100) {
 cout << n << endl;
 n++;
}
```

2.6.3 Do-While Loop

The do-while loop is a variant of the while loop. Unlike while loop, the do-while loop executes the code block at least once, before checking the ***condition***. It will then continue to loop through the code block as long as the ***condition*** is true.

Syntax:

```
do {
   // code block
} while (condition);
```

The code below prints the numbers 1 to 100.

```
int n = 1;
do {
   cout << n << endl;
   n++;
} while (n <= 100);
```

2.6.4 Break and Continue

The ***break*** statement is used to exit from a loop without executing the code written after this. Whereas, the ***continue*** statement is used to continue to the next iteration without executing the code written after this. The use of the break and continue statement can be well understood by the following example which prints 1 to 100 except 50.

```
for (int n = 1; n <= 1000; n++) {
   if (n == 50) {
      continue;
   }else if(n > 100){
      break;
   }
cout << n << endl;
}
```

Here, the first *if* condition checks if the number is 50 or not. If so, it continues to the next iteration without executing the code written after it. The *else if* condition checks whether the number is greater than 100 or not. If so, it comes out of the loop without executing the statements after it. So, it prints 1 to 100 without 50.

2.7 FUNCTION

A function is a block of statements that are executed whenever it is called. Usually, a function is written to perform a specific task. A function is defined once but can be used many times. This supports the code reusability.

2.7.1 Function Declaration and Definition

A C++ function has two parts; ***function declaration*** and ***function definition***. In the function declaration, we specify the *function name* along with its *parameters* and the *return type*. Whereas, in function definition, we define the *actual body of the function* along with the statements to be executed.

Function declaration:

```
int foo(int i, int j);
```

Function definition:

```
int foo(int i, int j){
   int sum;
   sum = mi + j;
   return i+j;
}
```

2.8 ARRAY

An array is a collection of elements of the same data type. The elements of an array are stored in contiguous memory locations, which make it easy to access the elements using an index. The index of an array starts with 0. Arrays make it possible to store many elements in a single variable.

2.8.1 Declaring an Array

The example below shows how to declare an array. The first line declares an array of 100 integers. Here, *int* is the type of the array, which states that it can only store integers. The 100 inside the [and] is the size of the array. That means, it can store maximum of 100 integers. Similarly, in the second line, an array of strings is declared with 4 strings "bca", "bsc", "mca", "msc".

```
int a[100];
string name[4]={"bca","bsc","mca","msc"};
```

2.8.2 Accessing and Changing the Elements of an Array

In C++, an array index always starts with 0. Using array indexing we can access the first element of an array ***name*** as ***name[0]***. Similarly, the fourth element can be accessed as ***name[3]***. The elements of the array at a particular index can also be changed in a similar way. The example below shows how to access and change the value of an element in the array.

```
// name is an array of 5 4$ elements
string name[4]={"bca","bsc","mca","msc"};
// prints the first and fourth element of the array
cout<<name[0]<<name[3];
// changes the third element of the array
name[2] = "MCA";
```

2.9 STRING

A string is an array of characters surrounded by double-quotes. Strings are generally used to store text data.

In C++, a string can be created either using the *old 'C' style* method or creating a new *string class object*. To use new style string, we must include the header <string> in the source code.

```
#include<string>
char myString[] = "Hello"; // 'C' style string
string myString2 = "World"; // 'C++' style string
```

2.9.1 Operations on String

2.9.1.1 String Concatenation

The strings can be merged using the + operator to create a new string. This operation is called concatenation.

```
string firstName = "Isaac";
string lastName = "Newton";
string Name = firstName + " " + lastName;
cout << Name; // This prints "Isaac Newton"
```

2.9.1.2 String Length

To length()/size() function of string returns the number of characters in a string.

```
string data = "A quick brown fox jumps over the lazy dog.";
cout << "The length of the data string is:" << txt.length();
cout << "The length of the data string is:" << txt.size();
```

2.9.1.3 Accessing and Changing the Characters of a String

The character of a string in any specific position can be accessed using the indexing operator []. Any character can also be changed using the indexing operator. Like an array, the indexing in the string also starts with 0.


```
string str = "Hello";
cout << str[0]; // this prints 'H'
str[0] = 'h';
cout << str; // Now, this prints 'hello'
```

2.10 ENCAPSULATION

In C++, the bundling of data members and member functions inside a class is called encapsulation. It is one of the main features of object-oriented programming.

```
class Square {
  public:
     int length;
     int area() {
     return length * length;
  }
};
```

In the above program, the **area()** function calculates the area of a square. The **area()** function needs the length of the square to calculate the area. So, the data member **length** and the **area()** function are bundled together in the Square class.

It is also possible to restrict a class member to be accessed from outside the class. In the program above if the access specifier of the length variable is changed to private or protected, the access to the length variable is restricted. This is called **data hiding**.

2.11 CLASS

In object-oriented programming, everything is related to classes and objects. A class is a user defined data type that is used as a "template" or "blueprint" for creating objects. Whereas, an object is an instance of that class. A class may have some *attributes/variables* and *methods/functions*. They are called **class members**.

2.11.1 Class Members

The variables declared/defined inside a class are called member variables. Whereas, the functions declared/defined inside a class are called member functions or methods. There are two ways to define a function that belongs to a class.

- Inside the class:
- Outside the class:

The program below shows how to define a function inside and outside a class.

```
class Car { // The class
private:              // Access specifier
   int weight;       // Variable
   string color;     // Variable
public:              // Access specifier
   void drive() {    // function defined inside the class
      // code block
   }
   void brake();     // function declared here
};
// function definition outside the class
void Car::brake() {
   // code block
}
int main() {
   Car myCar;        // Create an object of Car
   myCar.drive();    // Call the drive method
   myCar.brake();    // Call the brake method
   return 0;
}
```

2.11.2 Constructors

A constructor is a special non-static member function having the same name as the class. A constructor is called whenever an object of the class is created.

2.11.3 Overloading Constructors

A constructor may have zero, one or more parameters. So, more than one constructor can be created inside a class with different number/type of arguments. This is known as constructor overloading. Depending on the number of parameters, constructors can be of different types:

- If there is no constructor inside the class, a constructor, with no parameter, is created by the compiler. This is called a *default constructor*.
- If the user creates a constructor with no parameter, it is called a *no-argument constructor*.
- A constructor with one or more parameters is called a *parameterized constructor*.


```
class MyClass {    // The class
public:            // Access specifier
   MyClass() {     // No-argument constructor
      /            / code block
   }
   MyClass() {     // one-argument constructor
      // code block
   }
   MyClass(int i, int j){ // parameterized constructor
      // code block
   }
};
```

2.11.4 Object

An object can be created either statically on a stack or dynamically on a heap. To create an object of a class statically, we need to specify the class name, followed by the object name.

```
// obj1 is a statically created object of class MyClass
MyClass obj1;
```

The *new* keyword is used to create an object of a class on the heap. It is called a *dynamically created object*.

```
// obj2 is a dynamically created object of class MyClass
MyClass obj2 = new MyClass();
```

2.11.5 Member Initialization in Constructors

We can assign an initial value for a variable at the time of declaration in the class. Otherwise, we have to assign a value to a variable inside the constructor after defining it in the class. A value can be assigned to a variable in different ways as shown below.

```
int m; // default initialization
int n = 5; // initialization with assignment
int x( 6 ); // initialization using parenthesis
int y{ 7 }; // initialization using braces
```

2.11.5.1 Member Initializer List

Not every type of variable can be assigned inside a constructor. A **const** variable or a **reference** variable cannot be assigned a value once it is declared. So, in these cases C++ assign a

value to these variables using *member initializer list*. This concept can be well understood using an example.

```
class MyClass{
private:
   int m_1;
   const int m_Const;
public:
   // Initialize const member variable
   MyClass() : m_Const{2} // initializer list
   {
      // This is an assignment, not initialization
      m_1 = 10;
   }
};
```

2.11.6 'this' Keyword

In every member function, **this** keyword is used to refer to the current object that is calling the function. The 'this' pointer is only available in member functions and not in friend functions. Let us try the example below to understand the concept of this pointer.

```
#include <iostream>
using namespace std;
class Rectangle {
   private:
      int length;
      int breadth;
   public:
      // Constructor definition
      Rectangle(int length, int b) {
         // must use 'this->length', as the parameter
         // name is also same with the member variable
         this->length = length;
         // no need to use 'this' pointer,
         // as the parameter name is different
         breadth = b;
      }
```



```
      int area() {
         return length * breadth;
      }
      int bigger(Rectangle rect) {
         return this->area() > rect.area();
      }
};
int main(void) {
   // declare first rectangle object rect1
   Rectangle rect1(10, 15);
   // declare second rectangle object rect2
   Rectangle rect2(11, 14);
   if(rect1.bigger(rect2)) {
      cout << "rect1 is bigger than rect2!" <<endl;
   }else {
      cout << "rect1 is not bigger than rect2" <<endl;
   }
   return 0;
}
```

From the example above it is clear that 'this' pointer can be used:

- to refer the current class instance variable
- to pass the current object as a parameter to the method.

2.11.7 Static Members

The **static** members in a class are declared using the 'static' keyword. Declaring a member as static ensures that only one copy of that is created for the class. The static members are shared by all the objects of the class. As the static data members are part of the class and not of the objects, it is not initialized inside any constructor. Rather, they are initialized outside the class using the *scope resolution operator* (::). Similarly, a static member function is also related to the class directly. So, a static member function can be called by the class directly. Static member functions do not have access to 'this' pointer. A static member function can only access static data members of the class, not the non-static data members. But, a non-static member function can access both static and non-static data members. The concept of static members in a class can be well understood by an example.

```
#include<iostream.h>
using namespace std;
```

```
class MyClass {
  // static variable, count number of all function calls
  static int aCount;
  // non-static variable, count number of non-static function calls
  int nsCount;
  public:
    MyClass(){
      nsCount = 0;
    }
    // static method can access static member variable,
    // which is 'aCount' here
    static void callme1() {
      aCount++;
    }
    // non-static member function can access static
    // as well as non-static member variable
    void callme2() {
      aCount++;
      nsCount++;
    }
};
// static variable initialized, outside the class
int Rectangle::aCount = 0;
```

The program above declares two variables; one *static* and another *non-static*. The non-static variable is initialized inside the class constructor. But the static variable is initialized outside the class using the scope resolution operator. The *callme1* function accessed and increases the value of **aCount** variable only, as it can access only static variables. Whereas, the callme2 function accessed and increases the value of both **aCount** and **nsCount** variables. Because, being a non-static function it has access to both static and non-static variables.

2.11.8 Const Member Functions

The '**const**' keyword is used with a variable or with an object to make the variable or the object un-modifiable. The **const** data members are either initialized when declared or it is initialized using the *member initializer list*. A const data member can never be initialized inside the constructor.

Like const data members, a member function can also be const. It is known as a const member function. The const member functions are those member functions in which the calling object cannot be changed. A member function is declared as const to avoid any accidental change to the calling object.

A const member function can be called by a const object as well as a non-const object. Whereas, a non-const member function can only be called by a non-const object.

```
class MyClass {
  private:
    int i;
  public:
    MyClass(int n){
      i=n;
    }
    void modify() const {
    // ERROR! const member function cannot modify calling object
    i++;
  }
  void change(){
    i++;
  }
};
```

In the program above, the line 'i++;' inside modify will through an error. Because modify is a const member function so it cannot modify the calling object.

2.12 INHERITANCE

In object-oriented programming, inheritance is an important feature. A class can inherit attributes and methods from another class. The class that inherits the properties is called a derived or child class. Whereas the class from which the properties are inherited is called the base or parent class. To define a derived class, we use a class derivation list to specify the base classes. A class derivation list names one or more base classes and has the form:

```
class DerivedClass: <access-specifier> BaseClass
```

Here, *access-specifier* is one of public, protected, or private, and base-class is the name of a previously defined class. If the access specifier is not used, then it is private by default.

C++ supports both multiple inheritance and multilevel inheritance. When a class is derived from more than one base class, it is called **multiple inheritance**. A class can also be derived from a class, which is derived from another class. This is known as **multilevel inheritance**.

2.12.1 Access Control and Inheritance

We can summarize the different access types and how they are accessed: According to the Table 2.1, a member in the same class can access other private, protected, and public members. The members in the derived class can access protected, and public members. Whereas, the members of any unrelated class can access only the public members of another class.

Table 2.1: Access Control and Inheritance

Access	Public	Protected	Private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

A derived class *cannot* inherit the following methods of the base class:

- Constructors, destructors and copy constructors of the base class.
- Overloaded operators of the base class.

The friend functions of the base class.

2.12.2 Type of Inheritance

From a base class, a derived class can be inherited either through public, protected or private inheritance. Protected or private inheritance is hardly used, but public inheritance is commonly used. The **public inheritance** makes **public** members of the base class **public** in the derived class. The accessibility of the base class members from the derived class depends on the type of inheritance and it is shown in the table below.

Table 2.2: Accessibility of Base Class Members in Derived Class

Inheritance Type	Base Member	Derived Member
Public inheritance	public	public
	protected	protected
	private	No Access
Protected inheritance	public	protected
	protected	protected
	private	No Access
Private inheritance	public	private
	protected	private
	private	No Access

2.13 POLYMORPHISM

The literal meaning of **Polymorphism** is '**many forms**'. That means the same entity can act differently in different situations. Polymorphism is one of the basic features of object-oriented programming. Polymorphism can be of two types; **compile-time polymorphism** and **runtime polymorphism**. In C++, compile-time polymorphism is achieved using **function overloading** and **operator overloading**. Whereas, runtime polymorphism is achieved using **function overriding** with the help of **virtual functions**.

2.13.1 Function Overloading

A class can have multiple functions with the same name but with different parameters. The parameters can be different in terms of number, order, and type. This is known as *function overloading*. Function overloading doesn't depend on the return type of the functions.

```
class A{
   void add(int m, int n){
      cout<<(m+n);
   }
   float add(float m, float m){
      float sum;
      sum = m + n;
      return sum;
   }
   double add(double m, double m, double p){
      return (m+n+p);
   }
};
```

The class in the program above has three overloaded 'add' functions. The first two functions have two parameters, but the parameters are of different types. The last add function has the different number of parameters.

2.13.2 Operator Overloading

In C++, we can redefine or overload most of the built-in operators with user-defined types as well. For example, the + operator is used to add primitive data types. Using operator overloading we can overload the functionality of + operator, so that it can add two objects also. Overloaded operators are actually functions with special names. The keyword *operator* is followed by the *symbol* for the operator being defined. Like any other function, an overloaded operator can have a return type and a parameter list. A sample overloaded operator is given below.

```
Complex operator+ (const Complex&);
```

2.13.2.1 Operator Overloading as a Member/Non-member Function

A **unary operator** can be overloaded by either a **non-static member function with** no argument or a **non-member function with one argument**. Whereas, a **binary operator** can be defined by either a **non-static member function with one argument** or a **non-member function with two arguments.**

- **Some operators can be overloaded as a member function only:** The assignment operator (=), the function call operator (()), the subscript operator ([]) and the class member access operator (→) can be overloaded as a member function only.
- **Some operators can be overloaded as a non-member function only:** The insertion operator (≪) and extraction operator (≫) can only be overloaded as a non-member function. They are overloaded using friend functions.
- **Some operators can be overloaded both as a member and non-member function:** The operators, such as +, -, *, /, ++, - - etc. can be overloaded as a member as well as non-member function.

2.13.2.2 Some Operators that Cannot be Overloaded

The following operators take a name, rather than a value, as their second operand and provide the primary means of referring to members. So they cannot be overloaded.

:: (scope resolution operator)

. (member selection operator)

.* (member selection through a pointer to member)

The following 'named operators' cannot be overloaded because they report fundamental facts about their operands.

- *sizeof* (sizeof operator)
- *typeid* (type_info of an object)

The ternary conditional expression operator cannot be overloaded.

- ?: conditional operator

2.13.2.3 Complex Class with Overloaded Operators

The program below explains how to overload different types of operators in a class.

```
#include<iostream.h>
#include<alloc.h>
#include<stdlib.h>
#include<conio.h>
class Complex{
  private:
    double real; //real part
```



```
    double imaginary; //imaginary part
  public:
    friend ostream& operator<<(ostream& output, const Complex &);
    friend istream& operator>>(istream& input, Complex &);
    //constructor
    Complex(double r= 0.0, double i= 0.0);
    //addition
    Complex operator+(const Complex &) const;
    //subtraction
    Complex operator-(const Complex &) const;
    //multiplication
    Complex operator*(const Complex &) const;
    //assignment const Complex& operator=(const Complex &);
    //equivalent int operator==(const Complex &) const;
    //not equivalent
    int operator!=(const Complex &) const;
    //post increment
    Complex operator++(int);
    //pre increment const Complex& operator++();
    //subscript double
    operator[](int position);
    //function call
    const Complex& operator()(double r, double i);
    //overloading new and delete operators
    void* operator new (size_t);
    void operator delete (void *);
};
Complex::Complex(double r, double i) :real(r), imaginary(i){}
Complex Complex::operator+(const Complex &operand2) const{
  return Complex(real + operand2.real, imaginary +operand2.imaginary);
}
Complex Complex::operator-(const Complex &operand2) const{
  return Complex(real - operand2.real, imaginary - operand2.imaginary);
}
```

```
const Complex& Complex::operator=(const Complex &right){
   real = right.real; imaginary = right.imaginary; return *this;
}
Complex Complex::operator*(const Complex &operand2) const{
   return(real*operand2.real)-(real*operand2.imaginary),
      (real*operand2.imaginary)+(imaginary*operand2.real));
}
int Complex::operator==(const Complex &right) const{
   if((real == right.real) && (imaginary == right.imaginary))
      return 1;
   else
      return 0;
}
int Complex::operator!=(const Complex &right) const{
   if ((real != right.real) || (imaginary != right.imaginary))
      return 0;
   else
      return 1;
}
const Complex& Complex::operator++(){
   real+=1.0;
   return *this;
}
Complex Complex::operator++(int){
   Complex temp;
   temp.real=real;
   temp.imaginary=imaginary;
   real+=1.0;
   return temp;
}
double Complex::operator[](int position){
   if (position==0)
      return real;
   else
```



```
      return imaginary ;
}
const Complex& Complex::operator()(double r, double i){
   real=r;
   imaginary=i;
   return *this;
}
ostream & operator<<(ostream& output, const Complex &c){
   output << c.real << "+ i" << c.imaginary;
   return output;
}
istream & operator>>(istream& input, Complex &c){
   input >> c.real;
   input >> c.imaginary;
   return input;
}
void* Complex::operator new(size_t size){
   cout<<"Calling new operator"<<endl;
   void *ptr = malloc(size);
   return ptr;
}
void Complex::operator delete(void *ptr){
   cout<<"Calling delete operator"<<endl; free(ptr);
}
int main() {
   Complex c1(1,2);
   Complex c2(3,4);
   Complex c3;
   c3=c1+c2;
   cout<<c3<<endl;
   c3(5,6);
   cout<<c3<<endl;
   cout<<c3[0]<<endl;
   if(c1==c3) { cout<<"c1 and c3 are equal\n";
```

```
} if(c2!=c3) { cout<<"c1 and c3 are not equalu"; }
Complex c4(7,8);
Complex c5(9,10);
c4=c5++;
cout<<c4<<","<<c5<<endl;
c4=++c5;
cout<<c4<<","<<c5<<endl;
Complex *c6=new Complex(11,12);
delete c6;
return 0;
}
```

2.13.3 Function Overriding

Runtime polymorphism is achieved using function overriding. The member functions of a base class can be redefined by its derived classes with the same name and same return type. It is called **function overriding**.

2.13.3.1 Virtual Function

A function is called a virtual function if it is defined in the base class with the '**virtual**' keyword. The 'virtual' keyword must be used with the base class function. Otherwise, it may not be overridden. Virtual functions ensure that the function is overridden.

If a **non-virtual** function is called using a base class pointer, it will always call the base class function irrespective of what type of object it is pointing to. But, if the function is a virtual function then which function will be called that depends on the type of object assigned to the base class pointer. It can be well understood by an example.

```
#include<iostream>
using namespace std;
class Base{
  public:
  // non-virtual function
  void print(){
      cout<<"we are in base::print"<<endl;
  }
  // virtual function virtual
  void show(){
      cout<<"we are in base::show"<<endl;
```



```
  }
};
class Derived:public Base{
  public:
      // overriding non-virtual function
  void print(){
      cout<<"we are in derived::print"<<endl;
  }
  // overriding virtual function
  virtual void show(){
      cout<<"we are in derived::show"<<endl;
  }
};
int main(){
  // b is the base class pointer
  Base *b = new Base();
  // it will call the base class print
  b->print();
  // base class pointer pointing to derived class object
  b = new Derived();
  // print is a non-virtual function
  // so, even if the object of of Derived class
  // it will again call the base class print
  b->print();
  // b is the base class pointer
  b = new Base();
  // it will call base class pointer
  b->show();
  // base class pointer pointing to derived class object
  b = new Derived();
  // show is a virtual function
  // so, it will call the derived class show
  b->show();
  return 0;
```

```
}
```

2.13.3.2 Pure Virtual Function and Abstract Class

A pure virtual function is a virtual function without any function body. A class with at least one pure virtual function is called an **abstract class**. It is called an abstract class because no object can be created from this class. Moreover, a derived class must implement all the pure virtual functions of its base class. Otherwise, the derived class also becomes an abstract class. It is explained in the example given below.

An abstract class is used as a base class and acts as an interface for the derived classes.

```
// An abstract class
class MyClass{
public:
   // the function declaration is followed by =0
// for a pure virtual function
virtual void print() = 0;
virtual void show() = 0;
};
// It is also an abstract class, because, it has not
// implemented another pure virtual function 'show'
class A:public MyClass{
public:
   virtual void print(){
   }
};
// It is a concrete class, because, it has implemented
// all the pure virtual functions
class B:public MyClass{
   public:
   virtual void print(){
   }
   virtual void show(){
   }
};
int main(){
   MyClass obj; // ERROR! MyClass is an abstract class
```



```
   A objA;// ERROR! A is an abstract class
   B objB; // OK. B is not an abstract class
   return 0;
}
```

2.13.3.3 Virtual Base Class

In *multiple inheritance*, it is possible that multiple 'instances' of the same class appear in an inheritance hierarchy. It leads to redundancy and ambiguity. We can think of the following scenario.

```
#include<iostream>
using namespace std;
class Base {
public:
   void callme() { }
};
class A : public Base {
   void callme() { }
};
class B : public Base {
   void callme() { }
};
class MyClass : public A, public B {
};
int main() {
   Base *b = new MyClass();
   // Ambiguity! which callme() is to call?
   // A::callme() or B::callme()?
   b->callme();
   return 0;
}
```

If the base class is derived as a virtual base class, then there will be no ambiguity. We have to make the following changes in classes A and B to run the program described above.

```
... ... class A : virtual public Base {
void callme() { }
```

```
};class B : virtual public Base {
void callme() { }
};
```

2.14 NAMESPACE

A namespace is designed to overcome the difficulty of using functions, classes, variables with the same name available in different libraries. That means namespaces are used to organize codes into logical groups. So that name collisions can be prevented when our code includes multiple libraries. A namespace logically divides the global scope with different names.

2.14.1 Defining a Namespace

A namespace definition begins with the keyword 'namespace' followed by the name spacename as follows:

```
namespace MyNamespace {
   void func();
      cout<<"This is a function";
   }
   int var=10;
}
```

To call either a function or a variable in the namespace, prepend the namespace name as follows:

```
namespace_name::func();
namespace_name::var;
```

2.14.2 Use of Namespace

```
// first namespace
namespace namespace1{
   void function(){
      cout << "Inside namespace1" << endl;
   }
   int var=10;
}
// second namespace
namespace namespace2{
```



```
   void function(){
      cout << "Inside namespace2" << endl;
   }
   int var=20;
}
int main(){
   // calls function from first namespace 'namespace1'.
   namespace1:: function ();
   // calls function from second namespace 'namespace2'.
   namespace2:: function ();
   return 0;
}
```

2.14.3 The 'using directive' and 'using declaration'

This **'using'** directive tells the C++ compiler that the subsequent code is making use of names in the specified *namespace*. The *namespace* is thus implied for the following code.

```
// first namespace
namespace namespace1{
   void function(){
      cout << "Inside namespace1" << endl;
   }
   int var=10;
}
// second namespace
namespace namespace2{
   void function(){
      cout << "Inside namespace2" << endl;
   }
   int var=20;
}
using namespace namespace1;
int main (){
   // This calls function from first namespace
   function ();
```

```
    return 0;
}
```

The **using** declaration can be used to refer to a particular item within a namespace. For example, if the only part of the namespace1 that you intend to use is the variable var, you can refer to it as follows:

```
using namespace1::var;
```

2.15 TEMPLATES

A template is a blueprint for creating a generic class or a function. It involves writing code in a way that is independent of any particular type. A template is a blueprint or formula for creating a generic class or a function. So, templates are the foundation of generic programming. The programs below show how to use ***function template*** and ***class template***.

```
template <typename T>
T Max (T a, T b){
    return (a < b) ? b:a;
}
int main(){
    int i = 10;
    int j = 20;
    cout << "Max(i, j):" << Max(i, j) << endl;
    double d1 = 10.5;
    double d2 = 10.7;
    cout << "Max(d1, d2):" << Max(d1, d2) << endl;
    string s1 = "Hello";
    string s2 = "World";
    cout << "Max(s1, s2):" << Max(s1, s2) << endl;
    return 0;
}
template <typename T>
class Math{
    private:
        T x, y;
    public:
        Math(T m,T n) {
```



```
            x=m;
            y=n;
    }
    T Max(){
        return (x>y)?x:y;
    }
};
int main(){
    Test<int> a1(5,6);
    cout << "Max(5,6):" << a1.Max() << endl;
    Test<double> a2(5.5,6.6);
    cout << "Max(5.5,6.6):" << a2.Max() << endl;
    Test<string> a3("hello", "world");
    cout << "Max("hello","world"):" << a3.Max() << endl;
    return 0;
}
```

2.16 FRIEND FUNCTION AND FRIEND CLASS

The *private* members of a class *cannot* be accessed from outside the class. The protected members of a class can only be accessed by its derived classes. However, if a class or a function from outside the class wants to access private or protected members of the class, it has to be a *friend* of the class. In this scenario, the outside class is called a **friend class** and the outside function is called a **friend function**. The '*friend*' keyword is used to make a class or a function friend of another class. The friend declaration can be anywhere in the class irrespective of the private or public section.

2.16.1 Friend Function

A *friend* function can either be a **member function of another class or a global function**. The program below shows how to implement friend functions.

```
#include<iostream>
using namespace std;
class MyClass; // forward declaration
class MyFriendClass {
    public:
    void modify(MyClass& obj);
};
```

```
class MyClass{
   private:
      int i;
   public:
      MyClass(){
         i = 10;
      }
      void display(){
       cout<<i<<endl;
}
   // friend function is a global function
   friend void change(MyClass& obj);
   // friend function is a member function
   friend void MyFriendClass::modify(MyClass& obj);
};
// need to define member function here
void MyFriendClass::modify(MyClass& obj){
   obj.i = 30;
}
// global non-member function
void change(MyClass& obj){
   obj.i = 20;
}
int main(){
   MyClass obj;
   change(obj); // change through global function
   obj.display();// This line outputs 20
   MyFriendClass cls;
   cls.modify(obj);
   // change through member function
   obj.display();// This line outputs 30
   return 0;
}
```

OUTPUT :-


```
$20$
$30$
```

2.16.2 Friend Class

A friend class, like a friend function, can access private and protected members of other class in which it is declared as friend. All the member functions of the friend class become friend functions of the class where it is declared as friend.

```
#include <iostream>
using namespace std;
// forward declaration
class MyFriend;
class MyClass {
   private:
      int i;
      // MyFriend class is a friend class now
      friend class MyFriend;
   public:
      MyClass(){
      i = 10;
   }
};
class MyFriend {
   public:
      void modify() {
         MyClass obj;
            // As a friend class it can change
            // the private member data 'i'
         obj.i= 20;
      }
};
int main() {
   MyFriend obj;
   obj.modify();
   return 0;
```

```
}
```

2.17 POINTERS

The memory address of any variable or object can be fetched using the & operator. This address can be stored in a special type of variable. A **pointer** is a variable that can store the memory address of another variable or object.

```
int i = 10;
                // The i variable of type integer
cout << i <<endl; // prints the value of i
cout << &i <<endl;// prints the memory address of i (e.g., 0x6ffe1c)
```

The type of a pointer and the type of the variable or object should be the same. Because a pointer variable points to a data type of the same type. A pointer is created with the * operator.

```
MyClass obj;  // object obj is of type MyClass
MyClass *ptr = &obj; // A pointer ptr, stores the address of obj
// call the callme function using obj
obj.callme();
// call the callme function using ptr
ptr->callme();
// call the callme function using object stored in ptr
(*ptr).callme();
```

In the example above, we have used the pointer variable to get the memory address of a variable using the & operator. It is called the **reference operator (&)**. We have also used the pointer to get the value of the object by using the * operator. It is called a **dereference operator (*)**.

2.18 SOLVED QUESTIONS

1. _____ is a mechanism of reusing and extending existing classes without modifying them, thus producing hierarchical relationships between them.
   A. Static Binding
   B. Dynamic Binding
   C. Inheritance
   D. Virtual class

   Solution: (C). The capability of a class to derive properties and characteristics from another class is called Inheritance. Through inheritance, an existing class can be extended and reused.

2. When the compiler generates a class, function, or static data member from a template, it is referred to as _____.
   A. Template instantiation
   B. Template specialization
   C. Partial specialization
   D. Function specialization

   Solution: (A). The generation of a class, function, or static data member from a template is called template instantiation.

3. Virtual functions are primarily used in _____.
   A. Inheritance
   B. Operator overloading
   C. Encapsulation
   D. Data binding

   Solution: (A). The virtual functions can be utilized where parent-child relationship is there between the classes. So, this can be used in inheritance.

4. The data members in a class are usually declared as _____.
   A. Protected
   B. Public
   C. Private
   D. Both A and C

   Solution: (C). By default, the members of the class are declared as private.

5. What is the output of the following program?

```
#include<iostream>
using namespace std;
void f() {
  static int i;
  ++i;
  cout<<i<< " ";
}
main() {
  f();
  f();
  f();
}
```

   A. 1 1 1
   B. 0 0 0
   C. 3 2 1
   D. 1 2 3

   Solution: (D). The default value of a static variable is 0 and a static variable always retain its value between the function calls.

6. Which of the following operator cannot be overloaded?
   A. Addition
   B. Scope resolution operator
   C. Multiplication
   D. Division

   Solution: (B). The operators that cannot be overloaded are :: (scope resolution operator), (member selection operator), * (member selection through a pointer to member), size of operator, and typeid (type_info of an object).

7. Constructors are member functions of a class that have the same name as the _____.
   A. Class name
   B. Data member
   C. Class name
   D. Other class data member

   Solution: (C). A constructor is a member function of a class. The name of the constructor is same as the class name.

8. Which of the following shows the operator overloading feature in C++?
   A. Polymorphism

B. Inheritance
C. Message passing
D. Both A and B

Solution: (A). In C++, operator overloading is compile-time polymorphism.

9. If a function is declared virtual in its base class, you can still access it directly using the _____.
   A. Virtual Keyword
   B. Scope resolution Operator
   C. Indirection Operator
   D. Address Operator

Solution: (B). The scope resolution operator (::) is used to access the base class virtual member functions.

10. The functions of the derived class can access _____ members of the base class but not the _____ members of the base class.
    A. Public and protected, private
    B. Private and protected, public
    C. Private, protected
    D. Private, public and protected

Solution: (A). A derived class can access the public as well as the protected member functions of the base class. The private member functions can only be accessed within the base class itself.

11. Every function can have _____ or _____ arguments.
    A. No, only one
    B. One, maximum two
    C. One, any number of
    D. No, any number of

Solution: (C). A function can have zero or any number of arguments.

12. The destructor is used for _____.
    A. Initializing of variables
    B. Deallocation of memory
    C. Construction of variable
    D. All of the above

Solution: (B). A destructor is a member function of a class. It is called whenever an object of the class is destroyed. It is used to deallocate the memory or any other resources captured by the object before deleting the object.

13. _____ pointers are not modifiable.
    A. That
    B. This
    C. Indirection
    D. Address

Solution: (B). The 'this' pointer works only inside any member function. The 'this' pointer points to the current object that has called the function. So, 'this' pointer cannot be changed.

14. What does your class can hold?
    A. Data
    B. Functions
    C. Both A and B
    D. None of the mentioned

Solution: (C). A class can hold data as well as functions to handle those data.

15. _____ is automatically created when constructors are used.
    A. Object
    B. Destructor
    C. Array
    D. Reference

Solution: (B). The destructor is automatically created whenever a constructor is used.

16. When are the global objects are destroyed?
    A. When the control comes out of the block in which they are being used
    B. When the program terminates
    C. When the control comes out of the function in which they are being used
    D. As soon as local object die

Solution: (B). The global objects have global scope and they are accessible from any class or any function. So, the global objects are destroyed only when the program terminates.

17. _____ is used to make a copy of one class object from another class object of the same class type.
    A. Constructor
    B. Copy Constructor
    C. Destructor
    D. Copy Destructor

Solution: (B). As the name suggests, the copy constructor is used to make a copy of an object.

18. Data members which are static _____.
    A. Cannot be assigned a value
    B. Can be used in static or non-static member function
    C. Cannot be defined in a union
    D. Cannot be accessed outside the class

Solution: (B). A static data member is related to the class not to the object.

19. cout is a/an _____.
    A. operator
    B. function
    C. object
    D. macro

Solution: (C). *cout* is an object of 'ostream' class and *cin* is an object of 'istream' class.

20. We can overload which of the following C++ operators?
    A. Arithmetic operator (+, -, *, /)
    B. Class Member Access Operators (., .*)
    C. Size operator (sizeof)
    D. All of the above

Solution: (A). The arithmetic operators can be overloaded. Neither class member access operators nor the sizeof operator can be overloaded.

21. In the case of _____ function, arguments may be passed either by value or by reference.
    A. Private
    B. Friend
    C. Member
    D. Public

Solution: (B). A friend function is a non member function of a class which can access the private members of the class. In a friend function, arguments may be passed either by value or by reference.

22. When a destructor is called?
    A. After the end of object life
    B. Anytime in between object's lifespan
    C. At end of whole program
    D. Just before the end of object life

Solution: (D). The destructor is called just before the object is destroyed. This ensures that all the resources allocated for the object are freed before deleting the object.

23. Choose the correct sequence of destructors being called for the following code:

```
class A{
};class B{
};class C: public A, public B{
};
```

A. A(), B(), C()
B. B(), C(), A()
C. A(), C(), B()
D. C(), B(), A()

Solution: (D). In multiple inheritance, the base class constructors are called according to the sequence in the inheritance list. The destructors are called in the reverse order.

24. Destructors can be _____.
A. Abstract type
B. Virtual
C. Void
D. Any type depending on situation

Solution: (B). It's a good practice to keep the base class destructor as virtual. It ensures that the child class destructor is called before the base class destructor.

25. When is it advised to have user-defined destructor?
A. When a class contains some pointer to memory allocated in class
B. When a class contains static variables
C. When a class contains static functions
D. When a class is inheriting another class only

Solution: (A). Dynamically allocated memory can only be freed using the destructor. It prevents memory leaks, so it is advised to define a destructor when pointers are involved in class.

Answers:

1. C	2. A	3. A	4. C	5. D
6. B	7. C	8. A	9. B	10. A
11. C	12. B	13. B	14. C	15. B
16. B	17. B	18. B	19. C	20. A
21. B	22. D	23. D	24. B	25. A

2.19 PRACTICE QUESTION SETS

1. Which data members among the following are static by default?
A. extern
B. integer
C. const
D. void

2. Operator overloading is also called _____ polymorphism.
A. Runtime
B. Initial time
C. Compile time
D. Completion time

3. When overloading unary operator using Friend function, it requires _____ argument/s.
A. Zero
B. One

C. Two
D. None of these

4. While overloading binary operators using member function, it requires _____ argument/s.
A. Zero
B. One
C. Two
D. Three

5. >> is called as _____ operator.
A. Insertion
B. Extraction
C. Greater than
D. Lesser than

6. If in multiple inheritance, class C inherits class B, and Class B inherits class A. In which sequence are their destructors called if an object of class C was declared?
A. ~A() then ~B() then ~C()
B. ~C() then ~A() then ~B()
C. ~C() then ~B() then ~A()
D. ~B() then ~C() then ~A()

7. Multiple Inheritances is the process of inheriting a class from _____.
A. Single parent class
B. More than one child class
C. More than one parent class
D. Only one child class

8. If in multiple inheritance, class C inherits class B, and Class B inherits class A. In which sequence are their destructors called if an object of class C is deleted using a pointer of base class A?
A. ~A() then ~B() then ~C()
B. Only ~C() is called
C. ~C() then ~B() then ~A()
D. Only ~A() is called

9. In the above question, what will be the sequence if the base class A's destructor is virtual?
A. ~A() then ~B() then ~C()
B. ~A() then ~C() then ~B()
C. ~C() then ~B() then ~A()
D. ~B() then ~A() then ~C()

10. Which feature of OOP is exhibited by the function overriding?
A. Polymorphism
B. Encapsulation
C. Abstraction
D. Inheritance

11. Encapsulation and abstraction differ as_____.
A. Hiding and hiding respectively
B. Binding and Hiding respectively
C. Hiding and Binding respectively
D. Can be used any way

12. Copy constructor definition requires _____.
A. Object to be passed by value
B. Object not to be passed to it
C. Object to be passed by reference
D. Object to be passed with each data member value

13. The object _____.
A. Can be passed by reference
B. Can be passed by value
C. Can be passed by reference or value
D. Can be passed with reference

14. Where is the memory allocated for the objects?

A. Cache
B. ROM
C. HDD
D. RAM

15. What happens if non-static members are used in static member function?
A. Executes fine
B. Compile time error
C. Executes if that member function is not used
D. Runtime error

16. Which operator can be used to free the memory allocated for an object in C++?
A. Unallocate
B. Free
C. Collect
D. Delete

17. What exactly is passed when an object is passed by reference?
A. The original object name
B. The original object class name
C. The exact address of the object in memory
D. The exact address of data members

18. Which keyword among the following can be used to declare an array of objects in Java?
A. allocate
B. arr
C. new
D. create

19. What happens when an object is passed by reference?
A. Destructor is called at end of function

B. Destructor is called when called explicitly
C. Destructor is not called
D. Destructor is called when function is out of scope

20. The copy constructors can be used to _____.
A. Copy an object so that it can be passed to another primitive type variable
B. Copy an object for type casting
C. Copy an object so that it can be passed to a function
D. Copy an object so that it can be passed to a class

21. In which access should a constructor be defined, so that object of the class can be created in any function?
A. Any access specifier will work
B. Private
C. Public
D. Protected

22. Which among the following can show polymorphism?
A. Overloading &&
B. Overloading <<
C. Overloading ||
D. Overloading +=

23. How many types of access specifiers are provided in OOP (C++)?
A. 4
B. 3
C. 2
D. 1

24. What may be the name of the parameter that the template should take?

A. Same as template
B. Same as class
C. Same as function
D. None of the mentioned

25. How many types of templates are there in C++?
A. 1
B. 2
C. 3
D. 4

26. The abstract function definitions in derived classes is enforced at _____.
A. Runtime
B. Compile time
C. Writing code time
D. Interpreting time

27. Which of the following is not true about templates?
A. Template is a feature of C++ that allows us to write one code for different data types.
B. We can write one function that can be used for all data types including user defined types. Like sort(), max(), min(), etc.
C. We can write one class or struct that can be used for all data types including user-defined types like Linked List, Stack, Queue, etc.
D. Template is an example of compile time polymorphism.

28. What is the additional feature in classes that was not in structures?
A. Data members
B. Member functions
C. Static data allowed
D. Public access specifier

29. What is the general syntax for accessing the namespace variable?
A. namespace::operator
B. namespace.operator
C. namespace#operator
D. None of the mentioned

30. An object is _____.
A. One instance of a class
B. Another word for a class
C. A class with static method
D. A method that accesses a class

31. Identify the correct statement:
A. Namespace is used to group class, objects and functions.
B. Namespace is used to mark the beginning of the program.
C. Namespace is used to separate the class and objects.
D. None of the above

32. Which operator is used to signify the namespace?
A. Conditional operator
B. Ternary operator
C. Scope operator
D. None of the mentioned

33. What happens when an object is passed by reference?
A. Destructor is not called
B. Destructor is called at end of function
C. Destructor is called when function is out of scope
D. Destructor is called when called explicitly

34. The static data member _____.
A. Must be defined inside the class

B. Must be defined outside the class
C. Must be defined in main function
D. Must be defined using constructor

35. What is the new operator?
A. Allocates memory for an object or array
B. Allocates memory for an object or array and returns a particular pointer
C. Used as return type when an object is created
D. Used to declare any new thing in a program

36. Which among the following best defines abstraction?
A. Hiding the implementation
B. Showing the important data
C. Hiding the important data
D. Hiding the implementation and showing only the features

37. Which among the following can be viewed as combination of abstraction of data and code?
A. Class
B. Object
C. Inheritance
D. Interfaces

38. What happens when an object is passed by reference?
A. Destructor is not called
B. Destructor is called at end of function
C. Destructor is called when function is out of scope
D. Destructor is called when called explicitly

39. When copy constructor is called _____.
A. No object is created
B. A new object is created
C. Two objects are created
D. The original object is destroyed

40. Operators size of and _____?
A. Both can be overloaded
B. Both cannot be overloaded
C. Only size of can be overloaded
D. Only ?: can be overloaded

41. In C++, if new operator is used, when is the constructor called?
A. Before the allocation of memory
B. After the allocation of memory
C. Constructor is called to allocate memory
D. Depends on code

42. What does the following statement mean?
int (*fp)(char*)
A. Pointer to a pointer
B. Pointer to an array of chars
C. Pointer to function taking a char argument and returns an int
D. Function taking a char* argument and re-turning a pointer to int

43. If all the classes use private inheritance in multilevel inheritance, then _____
A. It will not be called multilevel inheritance
B. Each class can access only the private members of its parent
C. Each subsequent class can access all members of previous level parent classes

D. None of the members will be available to any other class

44. In which type is new memory location will be allocated?
A. Only in pass by reference
B. Only in pass by value
C. Both in pass by reference and value
D. Depends on the code

45. Which keyword is used to access the variable in namespace?
A. dynamic
B. const
C. static
D. using

46. Can abstract classes be used in multilevel inheritance?
A. Yes, always
B. Yes, only one abstract class
C. No, abstract class doesn't have constructors
D. No, never

47. It is _____ to define the abstract functions.
A. Mandatory for all the classes in program
B. Necessary for all the base classes
C. Necessary for all the derived classes
D. Not mandatory for all the derived classes

48. What is the output of the following program?

```
#include<iostream> using namespace std;
int n = 10;
int &f() {
    return n;
}
main() {
    f() = 20;
    cout<<n;
}
```

A. 10
B. Address of 'n'
C. Compile error
D. 20

49. Data members which will be inherited by the child class will have to be declared as _____.
A. Protected
B. Public
C. Private
D. Both A and B

50. What is the output of the following program?

```
#include<iostream> using namespace std;
class Base {
public:
  void display() {
    cout<<"[Base:dispay]";
  }
  virtual void print() {
    cout<<"[Base:print]";
  }
};
class Derived:public Base {
public:
  void display() {
    cout<<"[Derived:display]";
  }
  void print() {
    cout<<"[Derived:print]";
  }
```

70

```
};
main() {
  Base *p = new Derived();
  p->display();
  p->print();
}
```

A. [Base::display][Base::print]
B. [Derived::display][Derived::print]
C. [Derived::display][Base::print]
D. [Base::display][Derived::print]

Answers:

1. C	2. A	3. B	4. B	5. B
6. C	7. C	8. D	9. C	10. A
11. B	12. C	13. C	14. D	15. B
16. D	17. C	18. C	19. C	20. C
21. C	22. B	23. B	24. A	25. B
26. B	27. D	28. B	29. A	30. A
31. A	32. C	33. A	34. B	35. A
36. D	37. B	38. A	39. A	40. B
41. B	42. C	43. B	44. B	45. D
46. A	47. C	48. D	49. D	50. D

❐ ❐ ❐

CHAPTER 3

UNIX

SYLLABUS

Following Commands and its Different Options: ls, ps, pwd, mv, cp, touch, cat, time, cal, bc, sort, diff, wc, comm, ln, du, kill, sleep, chmod, chown, chgrp, top, nice, renice, cut, paste, grep, file, whereis, which, echo, env, PATH, CLASSPATH, find. vi editor, shell, wildcard, shell script.

3.1 SHELL

The shell is a UNIX/Linux is an interface between the user and the operating system. The shell provides an interface to accept commands into the system, executes the commands and displays the output of the commands. The two main components of the UNIX/Linux operating system are:

- Kernel
- Shell

A Kernel is at the heart of an operating system. It is used to communicate between the hardware and the operating system. The Kernel is the innermost part of an operating system, whereas Shell is the outermost part of it. A shell takes the input from the user in the form of commands. It then processes the command and gives an output. The shell provides an interface to the user so that the user can write programs, run commands and scripts. When the user runs the terminal, the shell issues a command prompt (usually $), where he can type the input. The command is then executed when the Enter key is hit. The output/result of the command is displayed on the terminal. As the name suggests, the shell wraps around the operating system to protect it from any damage. There are four types of shells available in UNIX/Linux. These are **Bourne shell, Korn shell, C shell** and **Bash shell** (available in Linux only).

3.2 SHELL SCRIPT

A shell script is a program that consists of commands. In a shell script, every line is actually a command. When a script is run, every command is executed line by line to produce the output/result. A shell script can be created by using the steps given below:

- Create a file using any editor
- Start the script with **#!/bin/sh**
- Write some code in the script
- Save the script with a filename say **myfile.sh**
- Executing the script file by typing *sh* **myfile.sh**

The operator "#!" is called the shebang operator. It directs the location of the interpreter by which the script is executed. Using "**#! /bin/sh**" in the script executes the script using **Bourne shell**.

3.3 VI EDITOR

The vi editor is the most popular and standard command-line text editor on the UNIX/Linux platforms. The vi editor was created by Bill Joy. Everything in vi is done through the keyboard.

There are two modes in vi; **insert/input mode** and **edit mode**. In insert/input mode user can enter some text into the file. Whereas, in edit mode user can move around the file, perform actions such as deleting, searching, replacing, copying, and saving a file.

Generally, to open the vi editor, the vi command is issued along with a filename. The vi command can be issued without the filename also. In that case, the file can be saved later. The vi editor always starts in edit mode. To write something in the file, the user has to switch to insert mode by pressing the i key. After writing into the file is completed, the *Esc* key is pressed to get back to the edit mode.

There are different ways of saving and exiting from the vi editor. The user needs to be in edit mode to save the file and exit from vi. The ways are given below:

- ZZ (capital ZZ) [used for saving and exit]
- :q! [used to discard all changes and exit]
- :w [used to save the file without exiting]
- :wq [used to save and exit]

Most of the commands in the vi editor are executed as soon as the user presses a sequence of keys. A command beginning with a : (colon) requires the user to hit enter key to complete the command.

In edit mode, the user can move around the file. Some of the commands are given below with their meaning:

- Up, down, left and right arrow keys: used to move the cursor around
- j, k, h, l: used to move the cursor down, up, left and right (as arrow keys)
- ^ (caret): places the cursor at the beginning of the current line
- $: places the cursor at the end of the current line
- *n*G: go to the *n*th line (*e.g.*, 5G moves the cursor to the 5th line)

- G: move the cursor to the last line
- b: move the cursor to the beginning of the previous word
- w: move the cursor to the beginning of the next word
- nw: move the cursor forward n word (*e.g.*, 5w moves the cursor five words forwards)
- nb: move the cursor back to n word
- {: move the cursor backward one paragraph
- }: move the cursor forward one paragraph

There are several ways to delete the contents of a file in the vi editor. Some of the commands are given below with their meaning:

- x [deletes a single character]
- nx [deletes n number of characters (*e.g.*, 3x deletes three characters)]
- dd [delete the current line]
- dn [d followed by a movement command (*e.g.*, d3w means delete 3 words)]

Undoing changes in vi is very easy. The character u is used to undo the last action. The character U (capital U) is used to undo all changes to the current line.

3.4 WILDCARD

Wildcards are a set of symbols that allow creating a pattern to match a set of files or directories. The very basic sets of wildcards are:

- [It represents zero or more characters.]
- ? [It represents a single character]
- [ ] [It represents a range of characters]

The use of wildcards can be well understood by the examples given in the table below:

Table 3.1: Use of Different Types of Wildcards

Wildcard	Meaning
$ ls a*	list all filenames starting with the letter a
$ ls a?z.sh	list all filenames starting with the letter a, ending with the letter z, and having any 1 character in between
$ ls s[aeiou]t.sh	list all filenames starting with the letter s, ending with the letter t, and having any one letter from a,e,i,o,u [written within the square brackets]
$ ls ???test*	list all filenames having at most 3 characters followed by test and ending with one or more occurrences of any character
$ ls [uerst]*	list all filenames having one or more occurrences of the letter u,e,r,s,t only

$ ls [abc][xyz]p*	list all filenames with any of the letters a,b,c as the first one, any of the letters x,y,z as the second one, followed by p, and finally ending with one or more occurrences of any character
$ ls test[0-9][!a-z][0-9]*	list all filenames that start with test, followed by any single-digit number, then followed by any character other than lowercase letters, followed by any digit with one or more occurrences

3.5 PATH AND CLASSPATH

PATH is an environmental variable in Linux and other Unix-like operating systems. The PATH variable consists of a set of paths to the directories which are searched for executable files/programs whenever a command is issued by a user. A CLASSPATH environment variable contains the location of user defined classes in Java.

3.6 ESSENTIAL UNIX COMMANDS

In this section, some important Unix/Linux commands have been discussed. The commands return 0 upon success, and some other integer value upon failure.

3.6.1 echo

The **echo** command is used to display a text/string on standard output.

Syntax: echo [option] [string]

Table 3.2: Options of echo Command

Options	Description
\e	allows escape sequences
\b	removes all spaces in between the texts
\c	suppress trailing text and new line
\n	creates new line
\t	create horizontal tab space
-n	omit trailing newline after the text

3.6.2 cd

The **cd** command is used to change the directory/folder.

Syntax: $cd [directory]

Table 3.3: Uses of *cd* command

Command	Meaning
$cd	change to the home directory
$cd ~	change to the home directory
$cd /	change to the root directory
$cd ..	change to the parent directory

$cd /home/user/Desktop	change to /home/user/Desktop directory
$cd Books\UNIX	change to subdirectory Books\UNIX
$cd "Hello World"	change to Hello World directory

3.6.3 pwd

The **pwd** command is used to print the current working directory.

Syntax: $pwd [option]

3.6.4 ls

The ls command lists files in the directory that match the name. If no name is given, it will list all files in the current directory.

Table 3.4: Options of *pwd* command

Options	Description
-L	Display the value of PWD from the Environment, even if it contains symbolic links
-P	Same as -L option. But for a symbolic link, the original pathname is displayed.

Syntax: ls [options] [names]

Table 3.5: Options of *ls* command

Options	Description
-a	Displays all files including . and ..
-A	Displays all files except . and ..
-d	Display only directories
-i	Displays the inode of each file
-R	Display subdirectories recursively
-l	Displays the long format listing
-r	Display files in reverse order
-s	Displays the allocated size (in blocks) of each file
-x	Displays entries by lines instead of columns
-X	Displays alphabetically by file extension
-c	Displays files by timestamp

3.6.5 chmod

The **chmod** command is used to change the access mode of a file.

Syntax: chmod [category][operation][permission] filename

Here, the category may be user(u), group(g), others(o), or all (*a* or *ugo*). The operation may be + (assigns permission), – (removes permission), or = (assign absolute permission). Similarly, the permission may be r (read permission), w (write permission), or x (execute permission).

3.6.6 chown

The **chown** command is used to transfer ownership of a file to another user. It can be used to change the group also.

Syntax: chown [options] user[:group] file(s)

The options may be one of the following:

Table 3.6: Options of *chown* command

Options	Description
-c	Reports when the ownership of the file is changed
-v	Shows the verbose information for every file processed
-f	Used to change the ownership forcefully

The following table shows different situations to change ownership.

Table 3.7: Different situations to change ownership

Option	Action
user (only user is given)	the user becomes the owner of the given files. The group ownership is not changed.
user: (user followed by :)	the user becomes the owner of the given files. The group ownership is changed to user's login group.
user:group (both user and group are given)	the user becomes the owner of the given files. The group ownership is changed to given group.
:group (: followed by group)	Only the group ownership is changed to the specified group.
: (only : is given)	Everything remain unchanged

3.6.7 chgrp

The **chgrp** command is used to change the group owner of a file.

Syntax: chgrp *options newgroup files*

- It is used to change the ownership of a file or a folder:

  chgrp *students myfile.c*

- It is used to recursively change the group ownership of all contents of a folder:

  chgrp -R *students myfolder*

- It is used to change the group of another file or folder depending on the group name of a reference file:

  chgrp -R *--reference=myfile.txt myfolder*

3.6.8 bc

bc is a character-based calculator in UNIX. Invoking bc command without argument provides a calculator environment. Arithmetic and other calculations can be performed in the environment.

For an example:

```
$bc
5 * 6
30
```

Here, bc multiplied two given numbers. We can also change the input base and output base in bc command.

```
$bc
ibase=2
1101
13
obase=16
12
E
```

The *ctrl+d* key is used to quit from the **bc** environment.

3.6.9 cal

The **cal** command is used to display the calendar of a year or a specific month.

The **cal** command by default displays the current month:

```
$ cal
```

January						
Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

cal -y : To display complete year
cal -m 12 2005 : Display the month 12th of 2005.
cal -w : will print week no.

3.6.10 cat

The **cat** (concatenate) command is used to read data from a file and gives the content of the file as output. It can also, be used to create a new file and concatenate the contents of multiple files together.

Table 3.8: Uses of *cal* command

Command	Action
cal -y	shows the whole current year
cal feb	shows the whole specific month
cal 2022	shows the whole specific year
cal 01 2021	shows a specific month in a specific year
cal -3	shows the previous, current and next month
cal -j	shows the current month of the Julian calendar

Syntax: cat <files>

Table 3.9: Uses of *cat* command

Command	Action
cat myfile1	displays the contents of myfile1
cat > myfile2	creates a new file named myfile2
cat myfile1 myfile2 > myfile3	merges the contents of myfile1 and myfile2 and send it to myfile3
cat *.c	displays the contents of all c files in the folder
cat >> myfile3	appends some contents to myfile3
cat -s myfile1	suppresses empty lines while displaying the file myfile1
cat -n myfile1	appends line number at the beginning while displaying myfile1
tac myfile1	display lines of myfile1 in reverse order

3.6.11 cp

The **cp** command copies a group of files.

Syntax: cp [option] <source file><dest file>|cp [option] <source files><dest directory>

Table 3.10: Uses of *cp* command

Command	Action
cp file1 file2	creates file2 from file1
cp file1 file2 file3 dir1	copies file1, file2, and file3 to dir1 directory

3.6.12 mv

The **mv** command is move to files and directories. It can also rename files and directories.

Syntax: mv [file1] [file2]

Table 3.11: *cp* command

Option	Meaning
-i	asks for confirmation
-r	recursively copies all subdirectories
-p	preserves attribute of a file
-backup	backup existing file before over-writing
-l	create a hard link of the file

UNIX

mv [file(s)] [directory]

Table 3.12: Uses of *mv* command

Command	Action
mv file1 file2	renames file1 to file2
mv file1 file2 file3 dir1	moves file1, file2, and file3 to dir1 directory

3.6.13 grep

The **grep** command finds a pattern in a file. It displays different outputs depending on the options used. By default, if no option is used, the **grep** command displays the lines containing the pattern.

Syntax: grep <options> [patterns] [filename(s)]

Table 3.13: Uses of *grep* command

Command	Action
grep hello file1	displays the lines of file1 containing the word 'hello'
grep 'hello world' file1 file2	displays the lines of file1 and file2 containing the word 'hello world'
grep -e 'hello | world' file1	displays the lines of file1 containing the word 'hello' **OR** 'world'
grep -e 'hello' -e 'world' file1	another way to display the lines of file1 containing the word 'hello' **OR** 'world'
grep -e 'hello'.*'world' file1	displays the lines of file1 containing both the words 'hello' **AND** 'world'

Table 3.14: *grep* command

Option	Meaning
-i	ignores case while matching pattern
-l	displays the filename where the pattern is found
-c	displays the count of lines containing the pattern along with the filename
-v	displays the lines not containing the pattern
-n	displays the line numbers containing the pattern

3.6.14 touch

The **touch** command is used to create a new file or to change the time-stamp of an existing file.

Syntax: touch <options> [expression] [filename(s)]

Table 3.15: Uses of *touch* command

Command	Action
touch myfile	creates myfile if myfile doesn't exists.
touch myfile	changes the modification and access time using the current time.

touch -t 202202081020 myfile	The time format here is YYYYMMDDhhmm.
touch -m -t 202202081020 myfile	changes the modification time of myfile
touch -a -t 202202081020 myfile	changes the access time of myfile

Table 3.16: Options of *touch* command

Option	Meaning
-m	used to change the modification time only.
-a	used to change the access time only.

3.6.15 ps

The ps command is used to view information regarding the processes. Ps stands for *process status*.

Syntax: ps [options]

Table 3.17: Uses of *ps* command

Command	Action
ps -e	displays information of all user and system processes
ps -l	displays long listing with memory-related information

Table 3.18: Options of *ps* command

Option	Meaning
-e	display information of all user and system processes.
-a	display information of all processes related to the current terminal.
-u <username>	display information of all processes related to the specific user.
-t <terminal>	display information of all processes related to the specific terminals.
-f	display full information including PPID.
-l	display long listing with memory related information.

3.6.16 comm

The comm command is used to compare two sorted files. It lists down differing entries in three different columns. The first column and the second column show the unique entries in the first file and the second file respectively. The third column shows the common entries in both files.

Syntax: comm <options> [file1[[file2]

Table 3.19: Options of *comm* command

Option	Meaning
-3	drops the third column/display the columns unique to each file
-1 -2	drops the first and second columns/display the column showing only the differences

3.6.17 diff

The **diff** command is used to display file differences and also tells what to do to make both the files identical.

Syntax: diff [file1] [file2]

3.6.18 cut

The **cut** command is used to cut a file vertically. The cut can be from one column to another column or from one field to another field.

Syntax: cut <option> [file]

Table 3.20: Uses of *cut* command

Command	Action
cut -c1-4 file1	displays columns 1 to 4 of file1
cut -c2,5 file1	displays columns 2 and 5 of file1
cut -f2-4 file1	displays field 2 to 4 of file1 [tab is the default field delimiter]
cut -d';' -f3,6-10 file1	displays fields 3, 5 and 6 to 10 of file1 [; is the delimiter]

Table 3.21: Options of *cut* command

Option	Meaning
-c	this is used to specify column numbers
-f	this is used to specify field numbers
-d	this is used to specify the field delimiter

3.6.19 du

The **du** command is used to display the disk usages by a directory taking all its subdirectories into account.

Syntax: du [directory(s)]

As the **du** command recursively displays disk usages of all the subdirectories, the display list can be very long sometimes. The **-s** option is used with the **du** command to display total disk usages by the directory instead of displaying all the subdirectories.

3.6.20 env

The **env** command is used to display all the environment variables in the system. The **export** command is used to set a variable. The command **export A=100** sets A as an environment varinble with a value 100.

Table 3.22: Options of *env* command

Option	Meaning
-i	temporarily clears all the environment variables for the current session
-u	removes variables from the environment

3.6.21 file

The **file** command is used to know the type of a file or a set of files. It identifies the type of file by context.

Syntax: file [file(s)]

For example, the output of the command 'file myfile.txt' is: myfile.txt: ASCII text and the output of the command 'file myfile2.html' is: myfile2.html: HTML Document, ASCII text.

3.6.22 ln

The **ln** command is used to create links. A link can be a soft link or a hard link. A soft link in UNIX is like a shortcut in Windows. A single copy of the original file is maintained in the file system, but more than one soft link of that file can be there. Deleting the original file makes the other soft links invalid.

In the case of a hard link, more than one file shares a single copy of the file in the file system. They share the same **i-node** number. Deleting any file does not make other files invalid. The file is deleted from the file system only if all the hard-linked files are deleted.

Syntax: ***ln*** myfile.txt newfile.txt

This command creates a file **newfile.txt** which is a hard link of the file **myfile.txt**.

Syntax: ***ln -s*** myfile.txt newfile.txt

The **-s** option ensures that the link created is a soft link or symbolic link.

3.6.23 kill

The **kill** command is used to kill or terminate a process prematurely. The kill command sends a signal to the process to terminate it.

Syntax: kill [options] [PID(s)]

Table 3.23: Options of *kill* command

Option	Meaning
-15	specifies signal number 15
-KILL	specifies signal name SIGKILL

3.6.24 nice

The **nice** command is used to run a job with a lower priority than its normal priority. The nice command is actually prefixed with a command.

Syntax: nice [-increment] command

Table 3.24: Options of *nice* command

Option	Meaning
-5	decrement priority by 5
3	increment priority by 3
-n5	decrement priority by 5, same as -5

3.6.25 renice

Unlike nice command, the **renice** command allows modifying the priority of a running process.

Syntax: renice [-n Increment] [-g |-p | -u] ID ...

Table 3.25: Options of *renice* command

Option	Meaning
-g	all IDs are interpreted as unsigned decimal integer process group IDs.
-p	all IDs are interpreted as unsigned integer process IDs.
-u	all IDs are interpreted as user names or user IDs.

3.6.26 paste

The **paste** command merges lines of files.

Syntax: paste [option] [file]

The **-d** option is used to specify a list of characters to be used as delimiters. By default, tab is used as the delimiter. The following command uses the = character as a delimiter instead of tab: *$paste* -d= **file1 file2**

3.6.27 pwd

The **pwd** command is used to print the current working directory. The pwd stands for 'Print Working Directory'.

Syntax: pwd [option]

Table 3.26: Options of *pwd* command

Option	Meaning
-L	prints the symbolic path
-P	prints the actual path

3.6.28 sleep

The **sleep** command is used to pause the execution for an amount of time.

Syntax: sleep number[suffix]

The **number** here is the amount in seconds to pause. Some suffix can be added to this number to make it minute, an hour or even day instead of seconds.

Table 3.27: Options of *sleep* command

Suffix	Meaning
s	treats the number as seconds
m	treats the number as minutes
h	treats the number as hours
d	treats the number as days

3.6.29 sort

The sort command is used to sort a file by arranging the lines of the file in a particular order.

Syntax: sort [option]... [file]...

Table 3.28: Options of *sort* command

Option	Meaning
-o	outputs the sorted result to a new file
-r	sorts in reverse order
-n	sorts a file numerically
-k	specifies the column number to sort
-c	checks whether the file is in sorted order
-u	remove duplicates after sorting

3.6.30 time

The **time** command is used to print a summary of real-time, user CPU time and system CPU time spent to execute a command.

Syntax: time [option] [command]

Table 3.29: Options of *time* command

Option	Meaning
-p	print time in POSIX format

3.6.31 top

The **top** command displays valuable system information like running processes, resource usage, processor activity and kernel tasks in real-time.

Syntax: top [option]

Table 3.30: Options of *top* command

Option	Meaning
-n	specifies the number of repetitions after which it stops
-u	display information of specific user process
-d	specifies delay time between screen updates

3.6.32 whereis

The **whereis** command, as the name specifies, is used to locate the binary, source and manual pages of a file. The whereis command searches the standard Unix locations for a specified command.

Syntax: whereis [options] file

Table 3.31: Options of *whereis* command

Option	Meaning
-b	it searches for binary files.
-m	it searches for manual sections of a file.
-s	it searches for the source of a file.

3.6.33 which

The **which** command is used to identify the location of a given executable file. The command searches for the executable file in the directories listed in the PATH environment variable.

Syntax: which [option] file(s)

Table 3.32: Options of *which* command

Option	Meaning
-a	print all matching pathnames

3.7 SOLVED QUESTIONS

1. Which is the core of the operating system?
   A. Shell
   B. Kernel
   C. Commands
   D. Script

   Solution: (B). The kernel is the heart of an operating system.

2. Which option of ls command used to view file i-node number?
   A. –l
   B. –o
   C. –a
   D. –i

   Solution: (D). The *-i* option in *ls* command displays the i-node number of files.

3. Find / -name '*' _____.
   A. List all files and directories recursively starting from /
   B. List a file named * in /
   C. List all files in / directory
   D. List all files and directories in / directory

   Solution: (A). The find command is used to find and list files and directories. The '-name *' option indicates all file names.

4. Applications communicate with kernel by using _____.
   A. System Calls
   B. C Programs
   C. Shell Script
   D. Shell

   Solution: (A). The system calls are system functions that are used to communicate with the kernel.

5. Which of the following is "NOT" a UNIX variant?
   A. Solaris
   B. AIX
   C. IRIX
   D. AS400

Solution: (D). The AS400 is from family of OS/400 operating system developed by IBM.

6. The system calls in UNIX is written using which language?
   A. C
   B. C++
   C. Assembly Language
   D. Fortran

Solution: (A). The core of the UNIX/Linux operating system is written using the C programming language.

7. Which of the following is considered as the super daemon in Unix?
   A. sysinit
   B. init
   C. inetd
   D. proc

Solution: (B). The init process is the very first process that starts during the booting of the computer system. **Init** is a daemon process that continues to run until the system is shut down.

8. Unix is which kind of Operating System?
   A. Multi-user
   B. Multi-processing
   C. Multi-tasking
   D. All of the mentioned

Solution: (D). The Unix/Linux is a multi-user, multi-processing and multi-tasking operating system.

9. What control character signals the end of the input file?
   A. ctrl + a
   B. ctrl + b
   C. ctrl + c
   D. ctrl + d

Solution: (D). In Unix/Linux, **ctrl + d** is the typical keyboard shortcut to end the input to the file. In Windows, it is **ctrl + z**.

10. Which among the following is used to write small programs to control Unix/Linux functionalities?
    A. Shell Commands
    B. Shell Script
    C. Filters
    D. C Language

Solution: (B). A shell script is a collection of Unix/Linux commands. When the shell script is run, the commands are executed one by one.

11. Which command changes a file's group owner?
    A. cgrp
    B. chgrp
    C. change
    D. group

Solution: (B). The 'chgrp' command is used to change the group owner of a file.

12. Which command is used to view compressed text file contents?
    A. cat
    B. type
    C. display
    D. print

Solution: (A). The **cat** command is used to create new files, view content of an existing file, concatenate files and redirect output in terminal or files.

13. Which command is used to extract a column from a text file?
    A. paste
    B. get
    C. cut
    D. tar

Solution: (C). The **cut** command is used to slice a line and extracts the text.

14. Which command is used to perform backup in Unix?
    A. backup
    B. cpio
    C. zip
    D. gzip

Solution: (B). The **cpio** stands for "copy in, copy out". It is used for taking backups in a *.cpio or *.tar file. This command can also copy files to and from the archives.

15. Which option of rm command is used to remove a directory with all its subdirectories?
    A. -b
    B. -o
    C. -p
    D. -r

Solution: (D). The -r option of **rm** command recursively deletes all subdirectories.

16. Which command is used to identify file type?
    A. type
    B. file
    C. finfo
    D. info

Solution: (B). The **file** command is used to know the type of a set of files. It identifies the type of file by context.

17. How to execute ls command inside a vi editor?
    A. !ls
    B. :ls
    C. :!ls
    D. We can't execute

Solution: (C). To run any command inside vi editor, the command should be followed by :! keystrokes.

18. Which command gives the first byte where the difference is in the file1 and file2?
    A. diff
    B. cmp
    C. comm
    D. ls -a

Solution: (B). The **cmp** command is used to compare two files of any type. If the files are different, the byte and line number at which the first difference occurred is reported.

19. Which command is used to set limits on file size?
    A. fsize
    B. flimit
    C. ulimit
    D. usize

Solution: (C). The **ulimit** command is used by the administrator to see, set, or limit the resource usage of any user.

20. Which option of rmdir command will remove all directories a, b, c if path is a/b/c?

A. -b
B. -o
C. -p
D. -t

Solution: (C). The -p option of **rmdir** command removes each of the directories in a path given as argument if they are already empty, starting from the last component.

21. Which of the following represents the root directory, home directory, current directory, and parent directory respectively?
A. / ~ . .
B. ~ . .. /
C. / ~ . ..
D. . .. / ~

Solution: (C). The / is the root directory, ~ is the home directory, . is the current directory, and .. is the parent directory in Unix/Linux.

22. Any file's attribute information is stored in which structure on the disk?
A. i-node
B. data blocks
C. file blocks
D. directory file

Solution: (A). The i-node structure of a file stores almost every important information about a file.

23. If two files on same partition point to the same i-node structure, they are called _____.
A. Soft links
B. Hard links
C. Alias
D. Special files

Solution: (B). A file and hard link of that file points to the same i-node structure. Hard links work in the same partition only.

24. Deleting a soft link _____.
A. Deletes the destination file
B. Deletes both the soft link and the destination file
C. Deletes just the soft link
D. Backup of the destination is automatically created

Solution: (C). A soft link of a file in Unix/Linux is like a shortcut file in Windows operating system. Deleting the soft link will not delete the original file.

25. Creation of hard links that point across partitions _____.
A. is allowed only to root user
B. can be done by all users
C. the effects are unspecified
D. is not allowed

Solution: (D). Hard links can be created only within the same partition.

Answers:

1. B	2. D	3. A	4. A	5. D
6. A	7. B	8. D	9. D	10. B
11. B	12. A	13. C	14. B	15. D
16. B	17. C	18. B	19. C	20. C
21. C	22. A	23. B	24. C	25. D

3.8 PRACTICE QUESTION SETS

1. What command is used to copy file and directories?
A. copy
B. cp
C. rn
D. cpy

2. How do you rename file "new" to file "old"?
A. mv new old
B. move new old
C. cp new old
D. rn new old

3. What UNIX command is used to update the modification time of a file?
A. time
B. modify
C. cat
D. touch

4. The file permission 764 means _____.
A. Everyone can read, group can execute only and the owner can read and write
B. Everyone can read and write, but owner alone can execute
C. Everyone can read, group including owner can write and the owner alone can execute
D. Everyone can read, write and execute

5. The permission – rwxr–r – represented in octal expression will be _____.
A. 777
B. 666
C. 744
D. 711

6. Effective group id can be set using following permission:
A. 0777
B. 2666
C. 4744
D. 1711

7. Sticky bit can be set using following permission:
A. 0777
B. 2666
C. 4744
D. 1711

8. If the umask value is 0002, what will be the permissions of new directory?
A. 777
B. 775
C. 774
D. 664

9. What will be output of following command:
$echo "The process id is" $$?
A. The process id is $$
B. The process id is $<pid>$<pid>
C. The process id is <pid><pid>
D. The process id is $$$$

10. How do you print the lines between 5 and 10, both inclusive?
A. cat filename | head | tail -6
B. cat filename | head | tail -5
C. cat filename | tail +5 | head
D. cat filename | tail -5 | head -10

11. The statement z = `expr 5 / 2` would store which of the following values in z?
A. 0
B. 1
C. 2
D. 2.5

12. The redirection 2> myfile implies _____.

A. Write file 2 to file myfile
B. Write standard output to myfile
C. Write standard error to myfile
D. None of the mentioned

13. cmd 2>&1 > abc will _____.
A. Write file2 to file1
B. Write standard output and standard error to abc
C. Write standard error to abc
D. Write standard output to abc and standard error to monitor

14. cmd > abc 2>&1 will _____.
A. Write file2 to file1
B. Write standard output and standard error to abc
C. Write standard error to abc
D. Write standard output to abc and standard error to monitor

15. Which command is used to close the vi editor?
A. q
B. wq
C. Both q and wq
D. None of the mentioned

16. Which vi editor command copies the current line of the file?
A. yy
B. yw
C. ye
D. None of the mentioned

17. Which command is used to delete the character before the cursor location in vi editor?
A. x
B. X
C. D
D. d

18. Which command searches the string in file opened in vi editor?
A. t or T
B. f or F
C. / or ?
D. None of the mentioned

19. Which command sets the number for all lines?
A. :set li
B. :set ln
C. :set nu
D. :set nl

20. When you use the ln command, which of the following occurs?
A. A file is created that points to an existing file
B. A file is created that is a copy of an existing file
C. A file is moved from one location to another
D. A file is renamed

21. Binary or executable files are _____.
A. Device files
B. Special files
C. Directory files
D. Regular files

22. The directory file contains _____.
A. Filenames and File Sizes
B. Filenames and Inode Numbers
C. Filenames and Address
D. Filenames and Permissions

23. Which of the following is not a valid file type on Linux?

A. Socket
B. Softlink
C. Inode
D. FIFO

24. Which are the two types of device files?
A. Character and Socket
B. Character and Block
C. Block and FIFO
D. Input and Output

25. Which of the following is not a valid run level?
A. S
B. 0
C. 8
D. 1

26. Which one of the following statements is not true?
A. vim editor is the improved version of vi editor
B. vi editor commands are not case sensitive
C. vi editor has two modes of operation: command mode and insert mode
D. vi stands for visual editor

27. Which command is used to close the vi editor?
A. q
B. wq
C. Both q and wq
D. None of the mentioned

28. Which command sets the number for all lines?
A. :set li
B. :set ln
C. :set nu
D. :set nl

29. Which command searches the string in file opened in vi editor?
A. / or ?
B. f or F
C. t or T
D. None of the mentioned

30. In vi editor, the key combination ctrl + f _____.
A. moves screen down one page
B. moves screen up one page
C. moves screen up one line
D. moves screen down one line

31. Which command shows all the abbreviations in vi editor?
A. ab
B. abb
C. Show
D. None of the mentioned

32. Which command is used to delete the character before the cursor location in vi editor?
A. X
B. x
C. D
D. d

33. What is a shell script?
A. group of commands
B. a file containing special symbols
C. a file containing a series of commands
D. group of functions

34. The first line in any shell script begins with a _____.

A. &
B. !
C. $
D. #

35. To spawn a child of our own choice for running the script, we can use _____ command.
A. ps
B. pr
C. sh
D. $$

36. Which command is used for taking inputs from the user in the scripts?
A. ip
B. input
C. read
D. write

37. What are positional parameters?
A. Special variables for assigning arguments from the command line
B. Pattern matching parameters
C. Special variables for reading user input
D. Special variables and patterns

38. The first argument is read by the shell into the parameter _____.
A. 1$
B. $0
C. $$
D. $1

39. Which one of the following statements is true?
A. Auto-indentation is not possible in vi editor
B. Auto-indentation can be set using the command ':set ai'
C. Auto-indentation can be set using the command ':set noai'
D. Auto-indentation is set by default in vi editor

40. Which vi editor command copies the current line of the file?
A. yy
B. yw
C. yc
D. None of the mentioned

41. Which of the following is used for storing the number of positional parameters?
A. $n
B. $#
C. $*
D. $2

42. Which of the following is not a wildcard in Unix/Linux?
A. *
B. ?
C. $
D. %

43. Which of the following files will not be deleted using "rm chap??"?
A. chap01
B. chap02
C. chaptd
D. chactd

44. Which of the following command will list all the hidden filenames in our directory having at least three characters after the dot (.)?
A. ls
B. ls -a
C. ls .???*
D. ls *

45. In vi editor, which command reads the content of another file?
A. read
B. r
C. ex
D. None of the mentioned

46. * and ? cannot match _____.
A. /
B. $
C. .
D. / and .

47. Which of the following files will not be listed using the following command:
ls chap0[1-4]?
A. chap02
B. chap05
C. chap01
D. chap04

48. Which of the following symbol is used for negating the character class?
A. .
B. *
C. !
D. %

49. Process which terminates before the parent process exits is known as _____.
A. Orphan
B. Zombie
C. Child
D. None of the above

50. The complete set of positional parameters is stored in _____ as a single string.
A. $n
B. $#
C. $*
D. $$

Answers:

1. B	2. A	3. D	4. C	5. C
6. B	7. D	8. B	9. C	10. A
11. C	12. C	13. D	14. B	15. C
16. A	17. B	18. C	19. C	20. A
21. D	22. B	23. C	24. B.	25. C
26. B	27. C	28. C	29. A	30. A
31. A	32. A	33. C	34. D	35. C
36. C	37. A	38. D	39. B	40. A
41. B	42. C	43. D	44. C	45. C
46. D	47. B	48. C	49. B	50. C

□ □ □

CHAPTER 4

DATA STRUCTURE

SYLLABUS

Searching, Sorting, Stack, Queue, Linked List, Tree, Graph.

4.1 CONCEPT OF DATA STRUCTURE

A data structure is a systematic way of organizing data in a computer so that it can be used effectively and efficiently. Different tasks use different data structures to minimize the time and space complexities. Data is growing day by day. So, to insert a new item or to search for an item, we need an efficient data structure that ensures that the whole process becomes fast.

4.1.1 Types of Data Structures

Data structures are classified mainly into two types:

(a) **Linear Data Structures:** In a linear data structure, all the elements are arranged in linear order. The elements are stored in a non-hierarchical manner where every element has its predecessors and successors except the first and the last element. Some of the popularly used linear data structures are **array, linked list, stack, queue etc.**

(b) **Non-Linear Data Structures:** In a non-linear data structure, each element could be connected with two or more other elements in a non-linear way to reflect a special relationship among them. As the elements are not arranged sequentially, all the data elements in the non-linear data structure cannot be traversed in a single run. Examples of non-linear data structures are **trees** and **graphs.**

4.1.2 Terms Related to Data Structure

- **Abstract Data Type (ADT):** Abstract Data Type (ADT) is a particular class of data structures or data types that have common features. The behaviour of the objects is defined by a set of values and a set of operations. They are defined only by the operations that may be performed on them but not how these operations will be implemented. An ADT does not specify how the data will be organized in memory or

what algorithms will be used for implementing the operations. As it gives an implementation-independent view of a data type, it is called Abstract Data Type.

4.2 STACK AND QUEUE

4.2.1 Stack

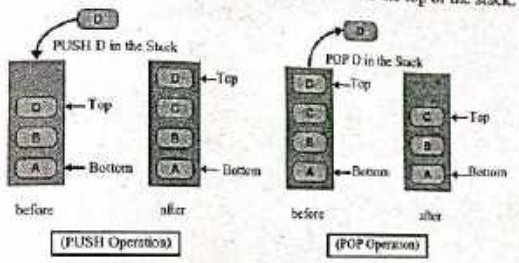
Stack is an ordered list in which, insertion and deletion can be performed only at one end that is called top. A stack is full when all the elements are inserted in the stack. Similarly, a stack is called an **empty stack** when no element is there in the stack. A stack is sometimes called as **Last-In-First-Out (LIFO)** list as the element which is inserted first in the stack, is deleted last from the stack.

4.2.2 Operations on Stack

A stack has two operations:

(a) **PUSH:** In the Push operation, an element is added to the top of the stack.

(b) **POP:** In the Pop operation, an element is taken from the top of the stack.



4.2.3 Applications

- Recursion
- Expression evaluations and conversions
- Parsing
- Browsers
- Editors
- Tree Traversals

4.2.4 Queue

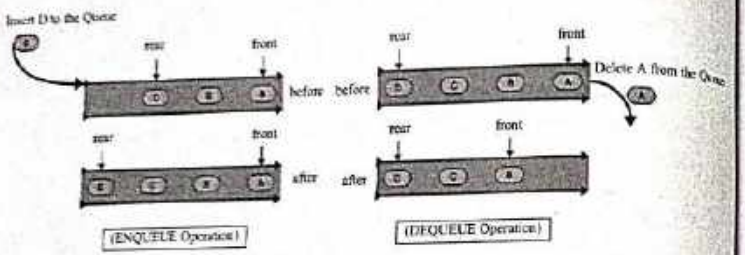
A queue can be defined as an ordered list that enables insert operations to be performed at one end called **REAR** and delete operations to be performed at another end called **FRONT**. A Queue is referred to be as **First In First Out (FIFO)** list as the element inserted first in the

queue is deleted first from the queue. An example of a queue is people waiting in line for a rail ticket.

4.2.5 Operations on Queue

A queue has two operations:

- **ENQUEUE:** In the Enqueue operation, an element is added to one side of the queue.
- **DEQUEUE:** In the Dequeue operation, an element is deleted from the other side of the queue.



4.2.6 Types of Queues

In data structure, a queue may be one of the following types:

- Simple Queue
- Circular Queue
- Priority Queue
- Deque (Double Ended Queue)

The simple queue is a normal queue where insertion takes place at the FRONT of the queue and deletion takes place at the END of the queue.

4.2.6.1 Circular Queue

- In a circular queue, the last node is connected to the first node.
- Circular queue is also called as Ring Buffer.
- Insertion in a circular queue happens at the FRONT and deletion at the END of the queue.

4.2.6.2 Priority Queue

- In a priority queue, the nodes will have some predefined priority.

- Insertion in a priority queue is performed in the order of arrival of the nodes.
- The node having the least priority will be the first to be removed from the priority queue.

4.2.6.3 Deque

In a Double Ended Queue (Deque), both insertion and deletion operations can be done at the FRONT and at the END of the queue.

4.2.6.4 Applications

- Queues are widely used as waiting lists for a single shared resource like printer, disk and CPU.
- Queues are used in the asynchronous transfer of data as in pipes, file IO and sockets.
- Queues are used as buffers in most of the applications.
- Queues are used in operating systems for handling interrupts.

4.3 LINKED LIST

4.3.1 Introduction

A linked list can be defined as a collection of objects called nodes that are randomly stored in the memory. A node contains at least two fields, *i.e.*, **data field** and link or **pointer field**. The data field stores the data at a particular address and the link field stores the address of the next node in the memory. The last node of the list contains a pointer to *null*.

4.3.2 Why Linked List is Used over Array?

Arrays can be used to store homogeneous data, but they have the following limitations:

1. The size of the arrays is fixed. So we must know the upper limit on the number of elements in advance."
2. Inserting a new element in an array is expensive because space must be created for the new element and the existing elements need to be shifted.

4.3.3 Drawbacks of Linked List

1. Random access is not allowed in a linked list. The elements need to be accessed sequentially starting from the first node.
2. Extra memory space is required for the pointer element for each node in the linked list.

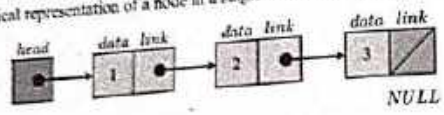
4.3.4 Types of Linked Lists

4.3.4.1 Singly Linked Lists

Simply, a list is a sequence of data, and a linked list is a sequence of data linked with each other. **Singly linked list is a sequence of elements in which every element has a link to its next element in the sequence.**

In any single linked list, the individual element is called a **"Node"**. **Every "Node" contains two fields, data and link**. The data field is used to store the actual value of that node and the link field is used to store the address of the next node in the sequence.

The graphical representation of a node in a single linked list is as follows:

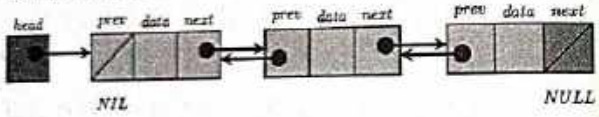


4.3.4.2 Doubly Linked Lists

Doubly linked list is a sequence of elements in which every element has a link to its previous and next element in the sequence.

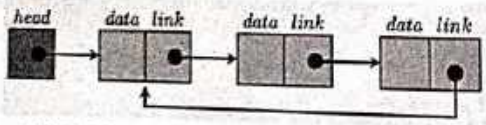
In any doubly linked list, **every "Node" contains at least three fields, one data and two link fields**. The data field is used to store the actual value of that node. One link field is used to store the address of the previous node and another link field is used to store the address of the next node in the sequence.

The graphical representation of a node in a doubly linked list is as follows:



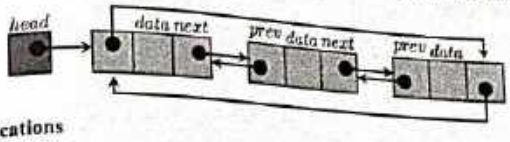
4.3.4.3 Circular Linked Lists

A circular linked list can be of two types; circular singly linked list and circular doubly linked list. In a circular singly linked list, the last node of the list contains a pointer to the first node of the list. We can have a circular singly linked list as well as a circular doubly linked list. We traverse a circular singly linked list until we reach the same node where we started. The circular singly liked list has no beginning and no ending. There is no *null* value present in the link part of any of the nodes.



A circular doubly linked list is a more complex type of data structure in which a node contains pointers to its previous node as well as the next node. The circular doubly linked list doesn't contain null in any of the node. The last node of the list contains the address of the first

node of the list. The first node of the list also contains address of the last node in its previous pointer.



4.3.5 Applications

In data structure and programming, linked lists are used almost everywhere because of the following reasons:

- It allocates the memory dynamically. All the nodes of a linked list are non-contiguously stored in the memory and linked together with the help of pointers.
- Sizing is no longer a problem since we do not need to define its size at the time of declaration. The list grows as per the program's demand and is limited to the available memory space.

4.3.6 Comparison of Sequential and Linked Data Structures

Sequential Data Structure Properties:

- **Contiguous:** An array takes up a block of memory, where each element in the array is situated beside another element within the array.
- **Random Access:** Because an array is contiguous, each element can be accessed directly by its index within the array.
- **Static:** An array takes up a static block of memory, containing its starting point, its size, and the value of each element. Because it is static, an array cannot grow or shrink. If an element is added to an array, a whole new array must be made.
- **Smaller memory allocation:** Because each element within an array only needs to store its value, compared to a linked list, an array takes up less memory.

Linked Data Structure Properties:

- **Non-contiguous:** Each node within a linked list does not have to be stored next to any other node within the linked list. A node can be stored in any free piece of memory.
- **Sequential Access:** Because a linked list is not contiguous, it does not support random access. To access a particular node within the linked list, the entire linked list must be traversed until the node is found.
- **Dynamic:** A linked list can be dynamically altered without a new linked list needing to be created. For example, a node can be added easily by inserting the node and adding a reference to it within the linked list.

- **Larger memory allocation:** Unlike an array, each node within a linked list needs to store both its value and a reference to another node; therefore, it takes up more memory.

Linked data structures are preferred over sequential data structures when the data is highly volatile and random access is not needed.

4.4 TREE

4.4.1 Introduction

In a linear data structure, data is organized in sequential order and in a non-linear data structure, data is organized in random order. The tree is the most popularly used non-linear data structure and it has a wide range of applications. A tree data structure can be defined as follows:

- Tree is a non-linear data structure that organizes data in a hierarchical structure and this is a recursive definition.
- Tree data structure is a collection of data (Node) which is organized in a **hierarchical** structure recursively.

In the tree data structure, every individual element is called as **Node.** A node in a tree data structure stores the actual data of that particular element and it is connected to another node by an edge in a hierarchical structure. In a tree data structure, **if we have N number of nodes then we can have a maximum of N-1 number of edges.** A tree has the following properties:

- The tree has one node called root. The tree originates from this, and hence it does not have any parent.
- Each node has one parent only but can have multiple children.
- Each node is connected to its children via edge.

4.4.2 Tree Terminologies

- **Root:** In a tree data structure, the first node is called the **root node.**
- **Edge:** The connecting link between any two nodes is called as an **edge.**
- **Parent:** The predecessor of any node is called its **parent node.**
- **Child:** The descendant of any node is called **its child node.**
- **Sibling:** The nodes which belong to the same parent are called as **siblings.**
- **Leaf:** The node which does not have a child is called a **leaf node.**
- **Internal Nodes:** The node having at least one child is an **internal node.**
- **Degree:** The total number of children of a node is the **degree** of that node.
- **Level:** In a tree, each step from top to bottom is called a **level** and the level count starts with '0' and incremented by one at each level.

- **Height:** The total number of edges from leaf node to a particular node in the longest path is called the **height** of that node.
- **Depth:** The total number of edges from the root node to a particular node is called the **depth** of that node.
- **Path:** The sequence of Nodes and Edges from one node to another node is called the **path** between that two nodes.
- **Subtree:** In a tree data structure, each child from a node forms a subtree along with its child nodes. Every child node will form a **subtree** on its parent node.

4.4.3 Binary Trees

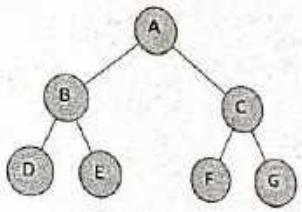


Fig. 4.1: Binary Tree

A binary tree is a special type of tree data structure in which every node can have a maximum of 2 children. One is known as the left child and the other is known as the right child. In other words, in a binary tree, every node can have either 0, 1 or 2 children but not more than 2 children.

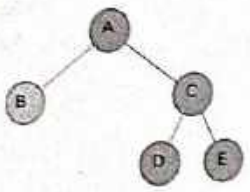
4.4.3.1 Types of Binary Trees

According to the property, a binary tree can be any of the following types:

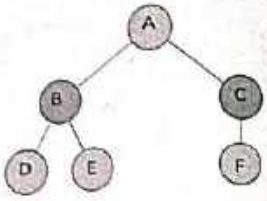
Strictly Binary Tree: A binary tree in which every node has either two or zero children is called Strictly Binary Tree.

Complete Binary Tree: A binary tree in which every internal node has exactly two children and all leaf nodes are at the same level except the last level.

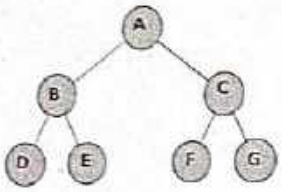
Full Binary Tree: A binary tree in which every internal node has exactly two children and all leaf nodes are at the same level is called Full Binary Tree.



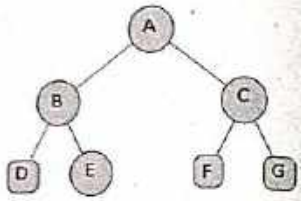
Strictly Binary Tree



Complete Binary Tree



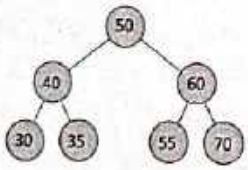
Full Binary Tree



Extended Binary Tree

Extended Binary Tree: An extended binary tree is a tree with some special nodes replacing every null subtree. Here, every regular node has two children, and every special node has no children.

4.4.3.2 Binary Search Tree



Binary Search Tree

A Binary Search Tree (BST) is a binary tree where each node contains a key. The left sub-tree of a node in a binary search tree contains nodes with keys less than the node's key whereas the right sub-tree of any node contains nodes with keys greater than or equals to the node's key. Binary Search Trees are used to implement different searching algorithms. They are also used for indexing and multi-level indexing. They are also helpful in storing sorted items.

4.4.4 Binary Trees Traversals

When we want to display a binary tree, we need to follow some order in which all the nodes of that binary tree must be visited. In any binary tree visiting order of nodes depends on the traversal method. The display order or visiting order of the nodes in a binary tree is called binary tree traversal. There are three types of binary tree traversals: **Pre-order Traversal, In-order Traversal** and **Post-order Traversal.**

4.4.4.1 Pre-order Traversal

In pre-order traversal, the root node is visited before the left child and the right child nodes. In this traversal, the root node is visited first, then its left child and later its right child [root-left-right]. This pre-order traversal is applicable for every root node of all sub-trees in the tree.

The pre-order traversal of the binary tree in Fig. 4.1 is **A-B-D-E-C-F-G.**

4.4.4.2 In-order Traversal

In the case of in-order traversal, the root node is visited between the left child and the right child. In this traversal, the left child node is visited first, then the root node is visited and later we go for visiting the right child node [left-root-right]. This in-order traversal is applicable for every root node of all subtrees in the tree. This is performed recursively for all nodes in the tree.

The in-order traversal of the binary tree in Fig. 4.1 is **D-B-E-A-F-C-G.**

4.4.4.3 Post-order Traversal

In Post-Order traversal, the root node is visited after the left child and the right child. In this traversal, the left child node is visited first, then its right child and then its root node [left-right-root]. This is recursively performed until the rightmost node is visited.

The post-order traversal of the binary tree in Fig. 4.1 is **D-E-B-F-G-C-A.**

4.4.4.4 Threaded Binary Trees

A binary tree is represented using array representation or linked list representation. When a binary tree is represented using a linked list, a NULL pointer is used instead if a node does not have a child. In any binary tree linked list representation, there are more numbers of NULL pointers than actual pointers. Generally, in any binary tree linked list representation, if there are 2N number of reference fields, then N+1 number of reference fields are filled with NULL (N+1 are NULL out of 2N). This NULL pointer does not play any role except indicating there is no link (no child).

A.J. Perlis and C. Thornton have proposed a new binary tree called "Threaded Binary Tree", which makes use of the NULL pointer to improve its traversal processes. In a threaded binary tree, NULL pointers are replaced by references to other nodes in the tree, called threads.

A threaded binary tree is also a binary tree in which all left child pointers that are NULL (in linked list representation) point to its in-order predecessor, and all right child pointers that are NULL (in linked list representation) point to its in-order successor.

4.5 SEARCHING AND SORTING

A searching algorithm is used to find an element from any given data structure. A sorting algorithm is used to arrange a set of elements, either in ascending or descending order, in any given data structure. Depending on the way of sorting the elements, sorting algorithms are categorized into the following types:

(a) **Internal and External Sorting:** A sorting is said to be an internal sorting if the entire sorting process takes place within the main memory of the computer. Insertion sort, Bubble sort, Selection sort and Heap sort are internal sorting. Some sorting algorithms can handle massive amounts of data for sorting. In this type of sorting, the data to be sorted do not fit into the main memory. They make use of the external memory (like a hard disk) of the computer to keep the data during sorting. This type of sorting is called **external sorting. Mergesort** is an external sort.

(b) **In-place and Not-in-place Sorting:** Some sorting algorithms do not need any extra space to sort the elements. They produce the output in the same memory that contains the original data. At most a small amount of extra memory may be required to keep some items at any time. These sorting algorithms are called in-place sorting. Otherwise, the sorting algorithm is a not-in-place sorting. Bubble sort, Selection sort, Insertion sort, Heap sort, and Quicksort are internal sorts. Merge sort is a not-in-place sorting.

(c) **Stable and Unstable Sorting:** The stability of a sorting algorithm depends on how it treats the equal elements during sorting. A sorting algorithm is called a stable sorting algorithm if it preserves the relative order of equal elements during sorting. Otherwise, the sorting algorithm is an unstable sorting algorithm. Merge sort, Insertion sort, Bubble sort are stable sorting algorithms. Whereas Quicksort, Heap sort and Selection sort are unstable sorting algorithms.

4.5.1 Linear Search

Linear search is a sequential way of searching an element from a given list. The search starts from the beginning of the data set, checks each item one after another until the element is found or the search ends. The searching algorithm works for both sorted and unsorted lists of elements. The complexity of linear search in $O(n)$.

4.5.2 Binary Search

Unlike linear search, binary search is a search technique that works only on a sorted list of elements. The binary search follows the divide-and-conquer approach to find an element from a list. At first, the search item is compared with the middle element of the list. If the match is found, the location of the middle element is returned. Otherwise, the search is continued in one of the halves depending on the search element is less than or greater than the middle element. The time complexity of the binary search is given in the table below.

Table 4.1: Time Complexity of Binary Search

Cases	Condition	Complexity
Best Case	The search element is all the middle of the list	$O(1)$
Average Case	The search element is anywhere except at the middle	$O(\log n)$
Worst case	The search element is not in the list	$O(\log n)$

4.5.3 Bubble Sort

The bubble sort is a simple comparison-based sorting algorithm that works by repeatedly swapping the adjacent elements if they are not in proper order. The time complexity of Bubble sort is given in the table below.

Table 4.2: Time Complexity of Bubble Sort

Cases	Condition	Complexity
Best Case	The array is already sorted, no need for sorting	$O(n)$
Average Case	The elements of the array are in jumbled order	$O(n^2)$
Worst case	Sort in ascending order when the elements are in descending order or *vice versa*	$O(n^2)$

4.5.4 Selection Sort

In Selection sort, the smallest element from an unsorted list is selected in each iteration and placed at the beginning of the unsorted list. The time complexity of Selection sort is given in the table below.

Table 4.3: Time Complexity of Selection Sort

Cases	Condition	Complexity
Best Case	The array is already sorted.	$O(n^2)$
Average Case	The elements of the array are in jumbled order	$O(n^2)$
Worst case	Sort in ascending order when the elements are in descending order or *vice versa*	$O(n^2)$

4.5.5 Insertion Sort

In Insertion sort, the first element is assumed to be already sorted. Then an unsorted element is selected. If the element is greater than the first element, it is placed on the right otherwise, to the left. In the same way, other unsorted numbers are taken and put in their right place. The time complexity of Insertion sort is given in the table below.

Table 4.4: Time Complexity of Insertion Sort

Cases	Condition	Complexity
Best Case	The array is already sorted	$O(n)$
Average Case	The elements of the array are in jumbled order	$O(n^2)$
Worst case	Sort in ascending order when the elements are in descending order or *vice versa*	$O(n^2)$

4.5.6 Merge Sort

Divide and Conquer Strategy: The divide and conquer strategy 'divides' a problem into smaller sub-problems. The sub-problems are solved first. Then the solutions to the sub-problems are 'combined' to get the result of the main problem. The merge sort and the quicksort algorithms are based on the divide and conquer strategy.

In merge sort, the array is divided into two halves until we get arrays of a single element. A single element array is always sorted. After that, the smaller arrays are combined to get larger sorted arrays. Thus the procedure is continued until the whole array is merged. The merge sort always follows the same procedure to sort an array irrespective of the order of elements in the array. So, the time complexity of merge sort is the same for all cases. The time complexity of quicksort is given in the table below.

Table 4.5: Time Complexity of Merge Sort

Cases	Complexity
Best Case	$O(nlogn)$
Average Case	$O(nlogn)$
Worst Case	$O(nlogn)$

4.5.7 Quicksort

In quicksort, an element from the array is picked as a pivot. The array is partitioned into two arrays. One array holds the values that are smaller than the pivot and the other array holds the values that are greater than the pivot. The pivot element is inserted in between these arrays. After that, the left and the right sub-arrays are also partitioned using the same approach recursively. This procedure is continued until a single element remains in the sub-arrays. The time complexity of quicksort is given in the table below.

Table 4.6: Time Complexity of Quicksort

Cases	Condition	Complexity
Best Case	When the pivot element is the middle element	$O(nlogn)$
Average Case	When the pivot element is anywhere in the array	$O(nlogn)$
Worst case	When the array is already sorted	$O(n^2)$

4.6 GRAPHS

4.6.1 Introduction

A graph data structure is a collection of nodes that are connected with each other. Each node contains some useful data. It can be visualized by using the following two basic components: **nodes and edges**. The relationship between the nodes is expressed using edges. If a graph comprises two nodes and an undirected edge between them, then it expresses a bi-directional relationship between the nodes and edge. An edge is a component that establishes a relationship between the nodes in a graph. An edge between two nodes expresses either a one-way or a two-way relationship between the nodes.

4.6.2 Types of Graphs

There are different types of graphs depending upon the number of edges, vertices, and interconnectivity. The types and their definitions are described in the table below.

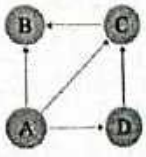
Type	Definition
Null Graph	A graph that contains no edges
Trivial Graph	A graph with a single vertex
Simple Graph	A graph having no parallel edges and no loops
Directed Graph	A graph where an edge has direction with an arrow
Complete Graph	A graph where every vertex is connected with every another vertex
Connected Graph	A graph in which at least one path exists between every pair of vertices
Regular Graph	A graph in which every vertex has same degree
Cyclic Graph	A graph that contains at least one cycle
Planar Graph	A graph that can be drawn on a plane without crossing any of the edges
Weighted Graph	A graph with edges having weights or numbers
Multi Graph	A graph having parallel edges and/or loops

4.6.3 Graph Representation

A graph can be represented in many ways. The type of representation depends on the type of operations to be performed, density and ease of use.

4.6.3.1 Adjacency Matrix

An adjacency matrix is a two-dimensional matrix of size V × V, where V is the number of vertices. In the adjacency matrix, each row and column represent a vertex. If there is a connection between a vertex i and a vertex j, the value of the element a[i][j] is taken as 1, otherwise, it is 0. A Graph and its corresponding adjacency matrix is given below:



	A	B	C	D
A	0	1	1	1
B	0	0	0	0
C	0	1	0	0
D	0	0	1	0

Fig. 4.4: A Graph and its Adjacency Matrix

4.6.3.2 Incidence Matrix

An incidence matrix is a two-dimensional matrix of size $V \times E$, where V is the number of vertices and E is the number of edges. In the incidence matrix, each row represents a vertex and each column represents an edge. This incidence matrix is filled with either 0 or 1 or −1. Where,

- 0 represents that the row vertex is not connected to the column edge.
- 1 represents that the row vertex has an outgoing column edge.
- −1 represents that the row vertex has an incoming column edge.

A Graph and its corresponding incidence matrix is given below:

4.6.3.3 Adjacency List

The adjacency list is a linked representation of a Graph. The index of the array represents a vertex and each element in its list represents the other vertices that are adjacent to the vertex. A Graph and its corresponding adjacency list is given below:

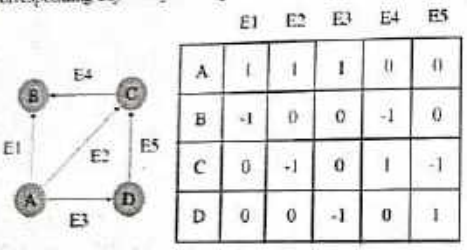


Fig. 4.5: A Graph and its Incidence Matrix

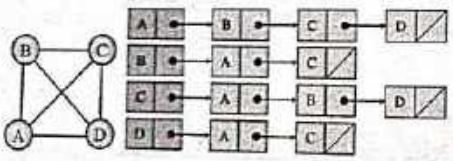


Fig. 4.6: A Graph and its Adjacency List

4.6.3.4 Traversal of a Graph

The process of visiting the vertices of a graph in a particular order is called graph traversal. The order of visiting the vertices is used to classify the traversals. There are mainly two types of graph traversals:

- **Breadth-First Search (BFS):** The BFS algorithm traverses all the nodes one by one at one level before moving to the next level. The traversal starts from the root node. The queue data structure is used to implement BFS algorithm.
- **Depth-First Search (DFS):** The DFS algorithm traverses the graph vertically. The traversal starts from the root node and traverses all the child nodes before moving to the adjacent nodes. The **stack** data structure is used to implement DFS algorithm.

4.6.3.5 Spanning Trees

A spanning tree of a graph is a tree that connects all the vertices of the graph. A **minimum spanning tree** (*MST*) is a tree that contains minimum possible number of edges. A spanning tree never contains a cycle and it is always connected. There are various algorithms available to find the minimum spanning tree in a graph. Two very popular algorithms for finding **MST** are:

- **Prim's Algorithm:** Prim's algorithm works by initialize the tree with any vertex of the graph. Then, minimum of the edges that connect any vertex of the tree to new vertices of the graph is added to the tree. The process is repeated until the **MST** is found.
- **Kruskal's Algorithm:** Kruskal's algorithm first sorts all the edges of the graph in ascending order. The minimum of the edges is taken and added to the tree. If adding the minimum edge creates any cycle, it is rejected. The process of adding next minimum edge to the tree is continued until the **MST** is formed.

4.6.4 Complexity of Data Structures, Sorting and Searching Algorithms

Algorithm	Time Complexity			Space Complexity
	Best Case	Average Case	Worst Case	Worst Case
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
Quick Sort	$O(n\log n)$	$O(n\log n)$		$O(\log n)$
Merge Sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	
Heap Sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(1)$
Shell Sort	$O(n\log n)$	$O(n\log^2 n)$	$O(n\log^2 n)$	$O(1)$
Bucket Sort	$O(n+k)$	$O(n+k)$		$O(n)$
Radix Sort	$O(nk)$	$O(nk)$	$O(nk)$	$O(n+k)$

Data Structure	Time Complexity								Space Complexity
	Best/Average Case				Worst Case				
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	Worst Case
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(1)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	O(1)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly Linked List	O(1)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly Linked List	O(1)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Binary Search Tree	O(log n)	O(log n)	O(log n)	O(log n)	O(n)	O(n)	O(n)	O(n)	O(n)
AVL Tree	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(n)
Red-Black Tree	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(n)
B Tree	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(n)
B+ Tree	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(n)

4.7 SOLVED QUESTIONS

1. Which data structure allows deleting data elements from front and inserting at rear?
   A. Stack
   B. Queue
   C. Deque
   D. Binary search tree

   Solution: (B). In queue data structure, deletion and insertion of elements are done at two different ends whereas in stack, deletion and insertion of elements are done at one end only.

2. Which of the following data structure is non-linear type?
   A. Strings
   B. Lists
   C. Queues
   D. Graphs

   Solution: (D). Trees and graphs are nonlinear data structures.

3. To represent hierarchical relationship between elements, which data structure is suitable?
   A. Deque
   B. Priority queue
   C. Tree
   D. All of the above

   Solution: (C). Hierarchical relationship can be represented by non-linear data structures only.

4. The following sequence of operation is performed on a stack:

   PUSH(10), PUSH(20), POP, PUSH(10), PUSH(20), POP, POP, POP, PUSH(20), POP

   The sequence of values popped out is _____.
   A. 20, 10, 20, 10, 20
   B. 20, 20, 10, 10, 20
   C. 10, 20, 20, 10, 20
   D. 20, 20, 10, 20, 10

   Solution: (B). A stack data structure is called a LIFO data structure. An element is pushed on top of another element and the top element is popped out first.

5. What do you call the selected keys in the quick sort method?
   A. Outer key
   B. Inner key
   C. Partition key
   D. Pivot key

   Solution: (D). The selected key in quick sort is called the pivot element.

6. Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree?
   A. 7 5 1 0 3 2 4 6 8 9
   B. 0 2 4 3 1 6 5 9 8 7
   C. 0 1 2 3 4 5 6 7 8 9
   D. 9 8 6 4 2 3 0 1 5 7

   Solution: (C). In-order traversal of a binary search tree always returns elements in sorted order.

7. The best data structure to check whether an arithmetic expression has balanced parentheses is a _____.
   A. queue
   B. stack
   C. tree
   D. list

   Solution: (B). The stack data structure is used to check proper parenthesizing.

8. Level order traversal of a rooted tree can be done by starting from the root and performing _____.
   A. Pre-order traversal
   B. In-order traversal
   C. Depth first search
   D. Breadth first search

   Solution: (D). The breadth first search is called level order traversal as all the elements in a level is traversed before the elements of the next level.

9. An abstract data type (ADT) is _____.
   A. Same as an abstract class
   B. A data type that cannot be instantiated
   C. A data type for which only the operations defined on it can be used, but none else.
   D. All of the above

   Solution: (C). ADT is user-defined data type that specifies the set of operations.

10. Post-order traversal of a given binary search tree T produces the following sequence of keys:

    10, 9, 23, 22, 27, 25, 15, 50, 95, 60, 40, 29

    Which one of the following sequences of keys can be the result of an in-order traversal of the tree T?
    A. 9, 10, 15, 22, 23, 25, 27, 29, 40, 50, 60, 95
    B. 9, 10, 15, 22, 40, 50, 60, 95, 23, 25, 27, 29
    C. 29, 15, 9, 10, 25, 22, 23, 27, 40, 60, 50, 95
    D. 95, 50, 60, 40, 27, 23, 22, 25, 10, 9, 15, 29

    Solution: (A). In-order traversal of a tree always returns elements in sorted order.

11. In a binary max-heap containing n numbers, the smallest element can be found in time _____.
    A. $O(n)$
    B. $O(\log n)$
    C. $O(\log \log n)$
    D. $O(1)$

Solution: (B). In a binary min-heap, the smallest element always exists at the root. In binary max-heap, the smallest element exists at the lowest level of the heap. So, to search the smallest element in a max-heap takes $O(\log n)$ time.

12. Which one of the following array represents a binary max-heap?
    A. 25, 12, 16, 13, 10, 8, 14
    B. 25, 14, 13, 16, 10, 8, 12
    C. 25, 14, 16, 13, 10, 8, 12
    D. 25, 14, 12, 13, 10, 8, 16

Solution: (C). In a binary max-heap, the maximum element is always at the root. All the parent elements are greater than its child elements. So, an element at the index i is greater than the elements at the index 2*i+1 and index 2*i+2.

13. Which of the following statements for a simple graph is correct?
    A. Every path is a trail
    B. Every trail is a path
    C. Every trail is a path as well as every path is a trail
    D. Path and trail have no relation

Solution: (A). In a graph, a trail is a walk with no repeated edge whereas a path is a walk with no repeated vertex. No repeated vertex implies no repeated edge. So, every path is actually a trail also.

14. What is the number of edges present in a complete graph having n vertices?
    A. (n*(n+1))/2
    B. (n*(n-1))/2
    C. n
    D. Information given is insufficient

Solution: (B). In a complete graph, every vertex is connected with every another vertex. So, the number of edges is (n-1) + (n-2) + (n-3) + ... + 1 = (n*(n-1))/2.

15. A connected planar graph having 6 vertices and 7 edges contain _____ regions.
    A. 15
    B. 3
    C. 1
    D. 11

Solution: (B). In a planar graph, the relation 'region = edge − vertex + 2' holds. So, number of regions = 7 − 6 + 2 = 3.

16. The number of elements in the adjacency matrix of a graph having 7 vertices is _____.
    A. 7
    B. 14
    C. 36
    D. 49

Solution: (D). An adjacency matrix is a two-dimensional matrix of size V × V, where V is the number of vertices. So, number of elements will be 7 × 7 = 49.

17. What is the content of the array after two delete operations on the correct answer to the previous question?
    A. 14, 13, 12, 10, 8
    B. 14, 12, 13, 8, 10
    C. 14, 13, 8, 12, 10
    D. 14, 13, 12, 8, 10

Solution: (D). Deletion can be done by following steps: (i) replace the root with last node, (ii) re-heapify the tree.

18. We are given a set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree?
    A. 0
    B. 1
    C. n!
    D. $(1/(n+1)).^{2n}C_n$

Solution: (B). In a binary search tree, the minimum value stays at leftmost node and the maximum value stays at rightmost node. Recursively, this is defined for all other nodes. So, there is only one way.

19. Which of the following is not true about spanning tree?
    A. It is a tree derived from a graph.
    B. All the nodes of a network appear on the tree only once.
    C. Spanning tree cannot have atmost two edges repeated.
    D. Spanning tree cannot be minimum and maximum.

Solution: (D). A spanning tree is any sub-graph of a given graph that does not have any cycle. Spanning tree could be minimum or maximum.

20. In a doubly linked list, the number of pointers affected for an insertion operation will be _____.
    A. 4
    B. 0
    C. 1
    D. Depends upon the nodes of the doubly linked list

Solution: (D). Let a node B is to be inserted after node A, and before node C. Then 'next' pointer of A, 'prev' pointer of C, and both 'prev' and 'next' pointers of B will be affected. So, total four pointers will be affected.

21. The number of different binary trees with 6 nodes is _____.
    A. 6
    B. 42
    C. 132
    D. 256

Solution: (C). The number of different binary trees with n vertices is (2n!)/[(n+1)!*n!]. So, total number of binary tree will be 12!/(7!*6!) = 132.

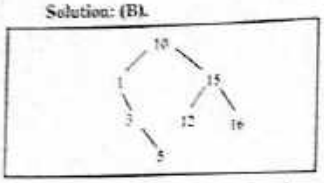
22. Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree?
    A. 7 5 1 0 3 2 4 6 8 9
    B. 0 2 4 3 1 6 5 9 8 7
    C. 0 1 2 3 4 5 6 7 8 9
    D. 9 8 6 4 2 3 0 1 5 7

Solution: (C). In-order traversal arranges the elements of a binary search tree in increasing order.

23. The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the height of the binary search tree (the height is the maximum distance of a leaf node from the root)?

A. 2
B. 3
C. 4
D. 6

Solution: (B).

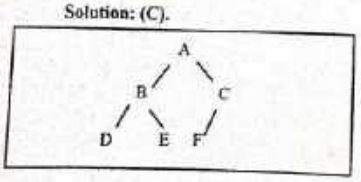


24. A binary search tree is used to locate the number 43. Which one of the following sequences is not possible?
A. 61 52 14 17 40 43
B. 2 3 50 40 60 43
C. 10 65 31 48 37 43
D. 81 61 52 14 41 43

Solution: (B). In a binary search sequence, all the numbers after a number n should be either greater than or less than the number n.

25. The post-order traversal of a binary tree is DEBFCA. Find out the pre-order traversal.
A. ABFCDE
B. ADBFEC
C. ABDECF
D. ABDCEF

Solution: (C).



So, the pre-order traversal will be ABDECF.

Answers:

1. B	2. D	3. C	4. B	5. D
6. C	7. B	8. D	9. C	10. A
11. B	12. C	13. A	14. B	15. B
16. D	17. D	18. B	19. D	20. A
21. C	22. C	23. B	24. B	25. C

4.8 PRACTICE QUESTION SETS

1. In a binary tree with n nodes, every node has an odd number of descendants. Every node is considered to be its own descendant. What is the number of nodes in the tree that has exactly one child?
A. 0
B. 1
C. (n – 1)/2
D. n – 1

2. Which of the following sorting algorithm is of divide-and-conquer type?
A. Bubble sort
B. Insertion sort
C. Quicksort
D. All of the above

3. The in-order traversal of which tree will yield a sorted listing of elements?
A. Binary trees
B. Binary search trees
C. Heaps
D. None of the above

4. A connected graph T without any cycles is called _____.
A. A tree graph
B. Free tree
C. A tree
D. All of the above

5. If every node u in G is adjacent to every other node v in G, a graph is said to be _____.
A. Isolated
B. Complete
C. Finite
D. Strongly connected

6. Merge sort is _____.
A. External sorting
B. Insertion sorting
C. Internal sorting
D. Exponential sorting

7. Two main measures for the efficiency of an algorithm are _____.
A. Processor and memory
B. Complexity and capacity
C. Time and space
D. Data and space

8. The time factor when determining the efficiency of algorithm is measured by _____.
A. Counting microseconds
B. Counting the number of key operations
C. Counting the number of statements
D. Counting the kilobytes of algorithm

9. Which of the following case does not exist in complexity theory?
A. Best case
B. Worst case
C. Average case
D. Complete case

10. The worst case occurs 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

11. The complexity of linear search and binary search algorithms are respectively _____.
A. $O(n)$ and $O(n)$
B. $O(n)$ and $O(\log n)$
C. $O(n^2)$ and $O(\log n)$
D. $O(n \log n)$ and $O(n \log n)$

12. The complexity of Bubble sort algorithm is _____.
A. $O(n)$
B. $O(\log n)$
C. $O(n^2)$
D. $O(n \log n)$

13. The post-fix expression for the infix expression A + B * (C + D)/F + D * E is _____.
A. AB + CD +* F/D + E*
B. ABCD + *F/DE *++
C. A * B + CD/F * DE++
D. A + * BCD/F * DE++

14. The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the height of the binary search tree (the height is the maximum distance of a leaf node from the root)?
A. 2
B. 3

C. 4

D. 6

15. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true?

I. 9679, 1989, 4199 hash to the same value.

II. 1471, 6171 hash to the same value.

III. All elements hash to the same value.

IV. Each element hashes to a different value.

A. I only

B. II only

C. I and II only

D. III or IV

16. A priority queue is implemented as a maxheap. Initially, it has five elements. The level-order traversal of the heap is given below: 10, 8, 5, 3, 2. Two new elements '1' and '7' are inserted in the heap in that order. The level-order traversal of the heap after the insertion of the elements is _____.

A. 10, 8, 7, 5, 3, 2, 1

B. 10, 8, 7, 2, 3, 1, 5

C. 10, 8, 7, 1, 2, 3, 5

D. 10, 8, 7, 3, 2, 1, 5

17. The following post-fix expression with single digit operands is evaluated using a stack:

823/23*+51* _____.

A. 6, 1

B. 5, 7

C. 3, 2

D. 1, 5

18. The in-order and pre-order traversal of a binary tree are 'd b e a f c g' and 'a b d e c f g', respectively. The post-order traversal of the binary tree is _____.

A. d e b f g c a

B. e d b g f c a

C. e d b f g c a

D. d e f g b c a

19. Which languages necessarily need heap allocation in the runtime environment?

A. Those that support recursion

B. Those that use dynamic scoping

C. Those that allow dynamic data structures

D. Those that use global variables

20. What is the outcome of the pre-fix expression +, -, *, 3, 2, /, 8, 4, 1?

A. 12

B. 11

C. 5

D. 4

21. Which of the following is not the correct statement for a stack data structure?

A. Arrays can be used to implement the stack

B. Stack follows FIFO

C. Elements are stored in a sequential manner

D. Top of the stack contains the last inserted element

22. Which of the following is the infix expression?

A. A+B*C

B. +A*BC

C. ABC+*

D. None of the above

23. Which one of the following is not the application of the stack data structure?

A. String reversal

B. Recursion

C. Backtracking

D. Asynchronous data transfer

24. When the user tries to delete the element from the empty stack, then the condition is said to be a _____.

A. Underflow

B. Garbage collection

C. Overflow

D. None of the above

25. Which of the following highly uses the concept of an array?

A. Binary search tree

B. Caching

C. Spatial locality

D. Scheduling of processes

26. Which of the following is the advantage of the array data structure?

A. Elements of mixed data types can be stored

B. Easier to access the elements in an array

C. Index of the first element starts from 1

D. Elements of an array cannot be sorted

27. How can we describe an array in the best possible way?

A. The Array shows a hierarchical structure

B. Arrays are immutable

C. Container that stores the elements of similar types

D. The Array is not a data structure

28. A graph is a tree if and only if graph is _____.

A. Directed graph

B. Contains no cycles

C. Planar

D. Completely connected

29. A vertex of in-degree zero in a directed graph is called a/an _____.

A. Root vertex

B. Isolated vertex

C. Sink

D. Articulation point

30. Minimum number of fields in each node of a doubly linked list is _____.

A. 2

B. 3

C. 4

D. None of the above

31. A graph in which all vertices have equal degree is known as _____.

A. Complete graph

B. Regular graph

C. Multi graph

D. Simple graph

32. Queues serve major role in _____.

A. Simulation of heap sort

B. Simulation of recursion

C. Simulation of limited resource allocation

D. Simulation of arbitrary linked list

33. Which one of the following is not the application of the queue data structure?

A. Resource shared between various systems
B. Data is transferred asynchronously
C. Load balancing
D. Balancing of symbols

34. In the linked list implementation of queue, where will the new element be inserted?
A. At the middle position of the linked list
B. At the head position of the linked list
C. At the tail position of the linked list
D. None of the above

35.
```
int fun(){
  if(isEmpty()) {
    return -10;
} else {
  int n;
  n= q[front]; front++;
  return n;
 }
}
```
Which operation is performed by the above code?
A. Enqueue
B. Deque
C. Return the front element
D. Both B and C

36. Which one of the following is the correct way to increment the rear end in a circular queue?
A. rear =rear+1
B. (rear+1) % max
C. (rear % max) + 1
D. None of the above

37. Which of the following determines the need for the circular queue?
A. Avoid wastage of memory
B. Access the queue using priority
C. Follows the FIFO principle
D. None of the above

38. Which one of the following is not the type of the queue?
A. Linear Queue
B. Circular Queue
C. Double Ended Queue
D. Single Ended Queue

39. Which of the following principle does Queue use?
A. LIFO principle
B. FIFO principle
C. Linear tree
D. Ordered array

40. What is another name for the circular queue among the following options?
A. Square buffer
B. Rectangle buffer
C. Ring buffer
D. None of the above

41. The minimum number of stacks required to implement a queue is ____.
A. 1
B. 3
C. 2
D. 5

42. Which of the following data structure is more appropriate for implementing quick sort iteratively?
A. Deque
B. Queue
C. Stack
D. Priority queue

43. A binary tree in which all its levels, except the last, have maximum numbers of nodes, and all the nodes in the last level have only one child it will be its left child. Name the tree.
A. Threaded tree
B. Complete binary tree
C. M-way search tree
D. Full binary tree

44. Which of the following data structure is required to convert arithmetic expression in in-fix to its equivalent post-fix notation?
A. Queue
B. Linked list
C. Binary search tree
D. None of the above

45. To perform level-order traversal on a binary tree, which of the following data structure will be required?
A. Hash table
B. Queue
C. Binary search tree
D. Stack

46. A parentheses checker program would be best implemented using _____.
A. List
B. Queue
C. Stack
D. Any of the above

47. The elements of a linked list are stored _____.
A. In a structure
B. In an array
C. Anywhere the computer has space for them
D. In contiguous memory locations

48. Which of the following piece of information does the data type to the compiler provide?
A. The way the data is to be interpreted
B. Range of values
C. Amount of memory a data element uses
D. All of the above

49. Which of the following statements is false?
A. A tree with n nodes has $(n - 1)$ edges.
B. A labelled rooted binary tree can be uniquely constructed given its post-order and pre-order traversal results.
C. A complete binary tree with n internal nodes has $(n + 1)$ leaves.
D. The maximum number of nodes in a binary tree of height h is $(2h + 1)$.

50. The number of edges in a complete graph of n vertices is_____.
A. $n(n+1)/2$
B. $n(n-1)/2$
C. $n^2/2$
D. N

Answers:

1. A	2. C	3. B	4. D	5. B
6. A	7. C	8. B	9. D	10. D
11. B	12. C	13. B	14. B	15. C
16. D	17. A	18. A	19. A	20. C
21. B	22. A	23. D	24. A	25. C
26. B	27. C	28. B	29. C	30. B
31. B	32. A	33. D	34. C	35. C
36. B	37. A	38. D	39. B	40. C
41. C	42. C	43. B	44. D	45. B
46. C	47. C	48. D	49. A	50. B

□ □ □

CHAPTER 5

INTRODUCTION OF COMPUTERS

SYLLABUS

Bus Structure, Basic I/O, Subroutines, Interrupt, DMA, RAM, ROM, Pipeline, System Calls.

5.1 BUS STRUCTURE

A bus is a **set of lines or wires** that connects multiple devices. Control signals and data are sent between the CPU and other components via buses. This is done to attain a decent operating speed. All peripherals in a computer system are connected to the microprocessor through a bus. There are three types of bus structures:

- **Address bus:** While reading from or writing into memory, the **address bus** carries the memory address. It can generate I/O post or device addresses from an I/O port. Only the CPU could send addresses over a unidirectional address bus, and other devices could not address the microprocessor. Computers nowadays have a bi-directional address bus.
- **Data bus:** The data is carried through a data bus. It is a bus that travels in both directions. It retrieves the instructions from memory and keeps track of the outcome of instruction. It can send commands to a controller or port for I/O devices. It can also carry data from a port or device controller.
- **Control bus:** There are different types of control signals in a computer system. This **memory read signal** and **memory write signals** are issued by the CPU or DMA controller when performing read or write operations with the memory, respectively. Whereas, the I/O read and I/O write signals are issued by the CPU when it is reading from or writing to an input or output port respectively. The **ready signal** is a special type of input signal generated by the CPU in order to synchronize the show I/O ports or memory with the fast CPU.

The most popularly used **USB** is a fast serial bus. It is used to connect an electronic device to a computer system. It is mostly used on personal computers, mobile phones, video games, etc.

5.2 BASIC I/O

An efficient way of communication between the central system and the outside environment in a computer is provided by a computer's I/O subsystem. It is in charge of the computer system's entire input-output activities.

5.2.1 Peripheral Devices

Peripheral devices are I/O devices that are connected to a computer. These devices, which are considered to be part of a computer system, are designed to read information into or out of the memory unit in response to commands from the CPU. Keyboards, mouses, printers, monitors and other display units are examples of peripheral devices. There are three types of peripherals:

- **Input peripherals:** These allow the computer to receive user inputs. Example: Mouse, Keyboard, etc.
- **Output peripherals:** These allow data to be sent from the computer to the user. Example: Monitor, Printer, etc.
- **Input-Output peripherals:** These allow both input and output. Example: Touch screen etc.

5.2.2 Modes of I/O Data Transfer

Data transfer between the central unit and I/O devices can be accomplished in one of three modes, as shown below:

- **Programmed I/O:** This method of data sharing between external devices and CPUs is the simplest. The Central Processing Unit (CPU), runs or executes a program that gives direct control of I/O activities in this method. The processor sends a command to the I/O module and waits for it to operate. In addition, the CPU continues to verify the state of the I/O module until the operation is completed. Its drawback is busy waiting, which means the CPU spends the majority of its time in a loop waiting for the I/O device to become available for usage.
- **Interrupt Driven I/O:** The CPU remains in the program loop in the programmed I/O approach until the I/O unit indicates that it is ready for data transfer. This is a time-consuming procedure because it wastes the processor's time. Interrupt driven I/O can be used to solve this problem. The interface creates an interrupt when it determines that the peripheral is ready for data transfer. When the CPU receives an interrupt signal, it stops what it's doing and handles the I/O transfer before returning to its prior processing activity.
- **Direct Memory Access (DMA):** The speed of data transfer would be improved by removing the CPU from the path and allowing the peripheral device to operate the memory buses directly. The interface transfers data to and from the memory via the memory bus in the DMA method. A DMA controller is responsible for data transport between peripherals and the memory unit. DMA is used by many hardware systems, including disc drive controllers, graphics cards, network cards, and sound cards, among others.

5.3 SUBROUTINES

In computer programming, it is a very common practice to perform a particular subtask several times with various input data. A **subroutine** is a term used to describe such a subtask. A subroutine could, for example, evaluate the factorial of a number given as an input or can check whether a given number is a prime number or not.

It is feasible to include the block of instructions that make up a subroutine at any point in the program where it is required. However, just one copy of the instructions that make up the subroutine is stored in memory to save memory. Any program that needs to use it simply branches to the beginning of the subroutine. This is known as **calling the subroutine** and the instruction that performs this is known as **Call** instruction. The program that calls the subroutine is known as **calling** program. The calling program first calls the subroutine. By performing a **return** instruction, the subroutine returns to the program that called it. The calling program must resume execution once a subroutine is returned, starting immediately after the instruction that called the subroutine. The **subroutine linkage method** is the means by which a computer allows to call and return from subroutines. The most basic technique of linking subroutines is to save the return address in a specified location, preferably in a register, known as **link register**, allocated to this function.

A special branch instruction, known as **Call instruction**, performs the following operations:

- Stores the contents of the **Program Counter (PC)**, which is the return address, in the **link register.**
- Transfer the control to the beginning of the subroutine. The last instruction of every subroutine commonly called return instruction.
- Transfers the program control to the address stored in the link register using the **return** instruction.

5.4 INTERRUPT

An interrupt is a signal. It requests the processor to suspend the execution of the current process. Then, the processor process and services the interrupt. The processor executes the corresponding interrupt service routine (ISR) to service the interrupt. After processing the interrupt, the processor resumes the execution of the suspended process.

5.4.1 Types of Interrupt

Interrupts can be of two types:

- **Hardware interrupts:** When the processor receives the interrupt request from an external I/O device, it is called a hardware interrupt. Hardware interrupts are of two types:

- **Maskable interrupt:** A maskable interrupt can be ignored or delayed to be processed if the CPU is busy executing a high priority process.
- **Non-maskable interrupt (NMI):** A non-maskable interrupt is immediately processed and serviced by the processor.

- **Software interrupts:** A software interrupt is caused by a system call or by an exceptional condition in the processor itself.

5.4.2 Interrupt Nesting

When the processor is busy executing an interrupt service routine, the interrupts are disabled so that no other device can raise another interrupt. This ensures that servicing of one interrupt is not interrupted by another interrupt. When the processor is executing an interrupt, if one more interrupt occurs again, then it is called a nested interrupt. When multiple devices raise interrupts simultaneously, the interrupts are serviced according to their priority. Whenever multiple devices interrupt a CPU at a time, and if the processor can handle them properly, it is said to have multiple interrupt processing ability.

5.5 DMA

There are three modes of data transfer between the memory and I/O devices; programmed I/O, Interrupt driven I/O, and **Direct memory access (DMA)**. In DMA mode, the block of data is transferred between the memory and I/O devices without the participation of the processor. The DMA mode of transfer is preferred over other modes due to their drawbacks. In programmed I/O and interrupt-driven I/O, no other activity is possible once the data transfer between the memory and I/O devices started. Because the processor is fully involved in transferring the data between memory and I/O devices. So, these are not effective ways of transferring a large block of data. But, the DMA controller can transfer a large amount of data at a faster rate without the involvement of the processor.

5.5.1 Types of DMA Transfer

The DMA controller can transfer the data in any of the three modes:

(a) **Burst Mode:** In burst mode, the DMA controller makes use of the system buses to transfer data. The CPU has to wait for the system buses until they are released by the DMA controller.

(b) **Cycle Stealing Mode:** In cycle stealing mode, the CPU is forced by the DMA controller to stop its normal execution and release the control over the bus. Then, the DMA controller uses the bus for transferring data. After transferring a byte of data, it releases the bus. The DMA controller again requests for the bus to the CPU for the transfer of next byte. In this way, the DMA controller steals a clock cycle of CPU for transferring every byte.

(c) **Transparent Mode:** In transparent mode, the DMA controller takes charge of the system bus only if it is not required by the CPU.

5.5.2 Advantages

- DMA reduces the clock cycle for writing or reading a block of data.
- DMA speeds up reading-writing tasks avoiding the involvement of the CPU.
- DMA reduces the overhead of the processor.

5.5.3 Disadvantages

- The cache coherence problem is the problem of synchronizing multiple local caches. The cache coherence problem can occur while using the DMA controller. It may access non-updated data from RAM if a proper synchronization mechanism is not provided.
- The DMA controller is a hardware unit. So, it is costlier to implement.

5.6 RAM AND ROM

Primary memory holds only those data and instructions on which the computer is currently working. It has a limited capacity and data is lost when power is switched off. That is why it is called volatile memory. It is divided into two subcategories RAM and ROM.

- **Random Access Memory (RAM):** RAM (Random Access Memory) is the internal memory of the CPU for storing data, programs, and program results. It is a read/write memory that stores data until the machine is working. As soon as the machine is switched off, data is erased. Data in the RAM can be accessed randomly. RAM is volatile, *i.e.*, data stored in it is lost when we switch off the computer. RAM is small, both in terms of its physical size and in the amount of data it can hold. RAM is of two types:
  - **Static RAM (SRAM):** The word static indicates that the memory retains its contents as long as power is being supplied. However, data is lost when the power gets down due to its volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not be refreshed regularly. The manufacturing cost of SRAM is high. SRAM is thus used as cache memory and has very fast access.
  - **Dynamic Ram (DRAM):** DRAM, unlike SRAM, must be continually refreshed in order to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory as it is cheap and small.
- **Read Only Memory (ROM):** ROM stands for Read Only Memory. The memory from which we can only read but cannot write on it. This type of memory is non-volatile. The information is stored permanently in such memories during manufacturing. A ROM stores such instructions that are required to start a computer. This operation is referred to as bootstrap. There are several types of ROM like PROM (Programmable Read Only Memory), EPRAM (Erasable and Programmable Read

Only Memory), EEPROM (Electrically Erasable and Programmable Read Only Memory), etc.

The difference between RAM and ROM is given in the following table.

Table 5.1: Difference between RAM and ROM

RAM	ROM
It is a volatile memory. It can retain data as long as the power is supplied.	It is a non-volatile memory. It can retain the data even when power is turned off.
Data stored in RAM can be accessed and modified.	Data stored in ROM can only be read.
It is used to store the data that is to be currently processed by CPU.	It stores the instructions required during bootstrap of the computer.
It is used in CPU Cache and in Primary memory.	It is used in Firmware, and in Microcontrollers.
It is a high-speed memory and costlier.	It is much slower and cheaper than RAM.

5.7 PIPELINE

Pipelining is used to execute multiple instructions simultaneously. It increases the throughput of the computer system. In pipelining, the instruction is divided into sub-tasks and each sub-task performs the dedicated task.

5.7.1 Types of Pipeline

Pipelines are divided into two categories:

- **Arithmetic Pipeline:** Arithmetic pipelines are used for floating-point operations, multiplication of fixed-point numbers, etc.
- **Instruction Pipeline:** In an instruction pipeline, a set of instructions can be executed by overlapping fetch, decode and execute phases of an instruction cycle. This increases the throughput of a computer. Using an instruction pipeline, multiple instructions can be executed simultaneously.

5.7.2 Advantages

- It increases the throughput of the computer system.
- It reduces the cycle time of the processor.
- It makes a reliable computer system.

5.7.3 Disadvantages

- It increases the instruction latency.
- The pipelined processor is complex to design and manufacture.

5.8 SYSTEM CALLS

The system call mechanism provides an interface between the process and the operating system. The system calls are the only way for a program to request any service from the kernel of the operating system.

5.8.1 How it Works?

At first, a process starts at user mode. When it tries to call a system call, the process starts to execute in kernel mode. The system call is executed in the kernel and when the execution is over, the process returns back to the user mode for execution.

5.8.2 Need for System Calls

A process needs a system call in the following situations:

- Creating, reading, writing and deleting a file
- Creating a new process or handling an existing process
- Sending or receiving packets through a network
- Handling I/O devices
- Accessing computer hardware devices

5.8.3 Types of System Calls

A system call can be any one of the following types:

- File Management
- Process Management
- Device Management
- Information Maintenance
- Communications

5.8.4 Some Examples of System Calls

Types	Windows	Linux
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()

Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

5.9 SOLVED QUESTIONS

1. RAM is of _____ types.
   A. 1
   B. 2
   C. 3
   D. 4

   Solution: (B). There are two types of RAM. They are SRAM (Static RAM) and DRAM (Dynamic RAM).

2. Which type of ROM is used for erasing purposes only?
   A. PROM
   B. EPROM
   C. EEPROM
   D. Both B and C

   Solution: (C). EEPROM stands for Electrically Erasable Programmable Read-Only Memory.

3. The DMA differs from the interrupt mode by _____.
   A. The involvement of the processor for the operation
   B. The method of accessing the I/O devices
   C. The amount of data transfer possible
   D. None of the mentioned

   Solution: (D). DMA is an approach of performing data transfers in bulk between memory and the external device without the intervention of the processor.

4. The DMA controller has _____ registers.
   A. 4
   B. 1
   C. 2
   D. 3

   Solution: (D). The Controller uses the registers to store the starting address, word count and the status of the operation.

5. When dealing with multiple devices interrupts, which mechanism is easy to implement?
   A. Polling method
   B. Vectored interrupts
   C. Interrupt nesting
   D. None of the mentioned

Solution: (A). In this method, the processor checks the IRQ bits of all the devices, whichever is enabled first. That device is serviced.

6. Which of the following groups are only input devices?
   A. Mouse, Keyboard, Monitor, Joystick
   B. Mouse, Keyboard, Printer, Light Pen
   C. Mouse, Keyboard, Scanner, Joystick, Light Pen
   D. Mouse, Keyboard, Trackball, Touch Screen, Microphone

   Solution: (D). Printer and Monitor are output device.

7. Which of the following is not a pointing device?
   A. Mouse
   B. Joystick
   C. Light pen
   D. Digitizer

   Solution: (D). Digitizer is an input device that converts analog information into a digital form.

8. The given hexadecimal number $(1E.53)_{16}$ is equivalent to _____.
   A. $(35.684)_8$
   B. $(36.246)_8$
   C. $(34.340)_8$
   D. $(35.599)_8$

   Solution: (B). $(1E.53)_{16}$ = $(0001\ 1110.0101\ 0011)_2$ = $(011\ 110.010\ 100\ 110)_2$ = $(36.246)_8$.

9. The octal equivalent of the decimal number $(417)_{10}$ is ____.
   A. $(64)_8$
   B. $(619)_8$
   C. $(640)_8$
   D. $(598)^8$

   Solution: (A). Octal equivalent of decimal number is obtained by dividing the number by 8 and collecting the remainder in reverse order.

10. On multiplication of (10.10) and (01.01), we get _____.
    A. 101.0010
    B. 0010.101
    C. 011.0010
    D. 110.0011

    Solution: (C). Use 0 * 0 = 0; 0 * 1 = 0; 1 * 0 = 0; 1 * 1 = 1.

11. 1's complement of 1011101 is _____.
    A. 0101110
    B. 1001101
    C. 0100010
    D. 1100101

    Solution: (C). 1's complement of a binary number is obtained by reversing the binary bits. All the 1's to 0's and 0's to 1's.

12. 2's complement of 11001011 is _____.
    A. 01010111
    B. 11010100
    C. 00110101
    D. 11100010

    Solution: (C). 2's complement of a binary number is obtained by finding the 1's complement of the number and then adding 1 to it. 2's complement of 11001011 = 00110100+ 1 = 00110101.

13. Binary coded decimal is a combination of _____.
    A. Two binary digits
    B. Three binary digits
    C. Four binary digits
    D. Five binary digits

    Solution: (C). Binary coded decimal is a combination of 4 binary digits. For example, 8421.

14. The parameter through which 16 distinct values can be represented is known as _____.
    A. Bit
    B. Byte
    C. Word
    D. Nibble

    Solution: (C). It can be represented up to 16 different values with the help of a Word. Nibble is a combination of four bits and Byte is a combination of 8 bits. It is "word" that is said to be a collection of 16-bits on most of the systems.

15. If the decimal number is a fraction, then its binary equivalent is obtained by _____ the number continuously by 2.
    A. Dividing
    B. Multiplying
    C. Adding
    D. Subtracting

    Solution: (B). On multiplying the decimal number continuously by 2, the binary equivalent is obtained by the collection of the integer part. However, if it is an integer, then it is binary equivalent is determined by dividing the number by 2 and collecting the remainders.

16. The largest two digit hexadecimal number is _____.
    A. $(FE)_{16}$
    B. $(FD)_{16}$
    C. $(FF)_{16}$
    D. $(EF)_{16}$

    Solution: (C). $(FE)_{16}$ is 254 in decimal system, while $(FD)_{16}$ is 253, $(EF)_{16}$ is 239 in decimal system, and $(FF)_{16}$ is 255. Thus, the largest two-digit hexadecimal number is $(FF)_{16}$.

17. The DMA differs from the interrupt mode by _____.
    A. The involvement of the processor for the operation
    B. The method of accessing the I/O devices
    C. The amount of data transfer possible
    D. None of the mentioned

    Solution: (D) DMA is an approach of performing data transfers in bulk between memory and the external device without the intervention of the processor.

18. The DMA transfers are performed by a control circuit called as _____.
    A. Device interface
    B. DMA controller
    C. Data controller
    D. Overlooker

    Solution: (B). The controller performs the functions that would normally be carried out by the processor.

19. After the completion of the DMA transfer, the processor is notified by _____.
    A. Acknowledge signal
    B. Interrupt signal
    C. WMFC signal
    D. None of the mentioned

    Solution: (B). The controller raises an interrupt signal to notify the processor that the transfer was complete.

20. MFC stands for _____.
    A. Memory Format Caches
    B. Memory Function Complete
    C. Memory Find Command
    D. Mass Format Command

    Solution: (B). This is a system command enabled when a memory function is completed by a process.

21. The addition of 2H and CH is _____.
    A. 2CH
    B. (2 + C)H
    C. EH
    D. None of the above

    Solution: (C). EH

22. The ASCII is expanded as _____.
    A. American Standard Code for Information Interchange
    B. American Stock Code for Information Interchange
    C. All Standard Code for Information Interchange
    D. All Standard Code for Information Interlink

    Solution: (A). American Standard Code for Information Interchange

23. Which of the following is not a positional number system?
    A. Roman Number System
    B. Octal Number System
    C. Binary Number System
    D. Hexadecimal Number System

    Solution: (A). Except roman number system all are positional number system.

24. Octal subtraction of $(232)_8$ from $(417)_8$ will give _____.
    A. 165
    B. 185
    C. 815
    D. 516

    Solution: (A). $(417)_8 - (232)_8 = (165)_8$

25. The two's complement in binary system is useful for expressing _____.
    A. Both positive and negative numbers
    B. Positive numbers
    C. Negative numbers
    D. None of these

    Solution: (C). The two's complement in binary system is useful for expressing negative numbers.

Answers:

1. B	2. C	3. D	4. D	5. A
6. E	7. D	8. B	9. A	10. C
11. C	12. C	13. C	14. C	15. B
16. C	17. D	18. B	19. B	20. B
21. C	22. A	23. A	24. A	25. C

5.10 PRACTICE QUESTION SETS

1. Which is a device that changes information into digital form?

A. Light pen
B. Modem
C. Mouse
D. Digitizer

2. RAM stands for _____.
A. Read Access Memory
B. Randomly Access Memory
C. Random Access Memory
D. Retains Access Memory

3. Which of the following types of memory is used for cache memory?
A. SDRAM
B. DRAM
C. SRAM
D. CRAM

4. What is the size of the computer accumulator register?
A. 4 bit
B. 4 KB
C. 4 bytes
D. 8 bytes

5. Which of the following is not the characteristic of Dynamic RAM?
A. Needs to be refreshed continuously
B. Smaller in size
C. Less power consumption
D. None of the above

6. In general, to have a well-defined learning problem, we must identity which of the following?
A. The class of tasks
B. The measure of performance to be improved

C. The source of experience
D. All of the above

7. UPS stands for _____.
A. Uninterruptible Program System
B. Uninterruptible Physical System
C. Uninterruptible Power System
D. Uninterruptible Power Storage

8. When we execute a C program, CPU runs in _____ mode.
A. User
B. Kernel
C. Supervisory
D. System

9. In _____ mode, the kernel runs on behalf of the user.
A. Kernel
B. User
C. Real
D. All

10. Match the following:

(Group-1)	(Group-2)
(i) Byte	(a) 8 bits
(ii) Nibble	(b) 16 bits
(iii) Word	(c) 4 bits

A. (i) → (c), (ii) → (b), (iii) → (a)
B. (i) → (a), (ii) → (b), (iii) → (c)
C. (i) → (b), (ii) → (c), (iii) → (a)
D. (i) → (a), (ii) → (c), (iii) → (b)

11. The primary memory (also called main memory) of a personal computer consists of _____.
A. ROM only
B. RAM only
C. Cache Memory
D. Both RAM and ROM

12. The Boot sector files of the system are stored in which computer memory?
A. RAM
B. Cache
C. ROM
D. Register

13. RAM is _____ and _____.
A. Volatile, temporary
B. Non-volatile, temporary
C. Volatile, permanent
D. Non-volatile, permanent

14. Which of the following is the lowest in the computer memory hierarchy?
A. Cache
B. RAM
C. Secondary memory
D. CPU registers

15. Which memory acts as a buffer between CPU and main memory?
A. RAM
B. ROM
C. Cache
D. Storage

16. Which of the following is used to transfer data between the processor (CPU) and memory?
A. Cache
B. TLB
C. Buffer
D. Registers

17. Which computer memory chip allows simultaneous both read and write operations?
A. ROM
B. RAM

C. PROM
D. EEPROM

18. When a machine is pipelined, the _____ execution of instructions requires pipelining of the functional unit.
A. Overloaded
B. Overrode
C. Overlapped
D. Overcrowded

19. Hazards in pipelines can make it necessary to _____ the pipeline.
A. Stall
B. Stake
C. Storm
D. None of the above

20. Sequential computer was improved from bit-serial to _____.
A. Bit-parallel
B. Byte-serial
C. Word-parallel
D. None of the above

21. Pipeline increases _____ of the processor.
A. Throughput
B. Storage
C. Predictivity
D. Latency

22. _____ is the logical structure of a computer's Random Access Memory (RAM).
A. Memory addressing
B. Operation field
C. Address field
D. Addressing mode

23. All UNIX and LINUX systems have one thing in common which is _____.

A. Set of system calls
B. Set of commands
C. Set of instructions
D. Set of text editors

24. The chmod command invokes the ____ system call.
A. chmod
B. ch
C. read
D. change

25. What invokes the system calls?
A. A privileged instruction
B. An indirect jump
C. A software interrupt
D. Polling

26. Which is the system call that is responsible for sending of SYN packets?
A. Connect
B. Bind
C. Socket
D. Listen

27. Runtime support system is the system that is provided by _____.
A. System call routines are mostly written in
B. System programs
C. System calls interface
D. Processes

28. System call routines of the operating system are mostly written in _____.
A. C
B. C++
C. Java
D. Both A and B

29. In DMA transfers, the required signals and addresses are given by the _____.
A. Processor
B. Device drivers
C. DMA controllers
D. The program itself

30. The interrupt servicing mechanism in which the requesting device identifies itself to the processor to be serviced is _____.
A. Polling
B. Vectored interrupts
C. Interrupt nesting
D. Simultaneous requesting

31. Which table handle stores the addresses of the interrupt handling sub-routines?
A. Interrupt-vector table
B. Vector table
C. Symbol link table
D. None of the mentioned

32. To extend the connectivity of the processor bus, we use _____.
A. PCI bus
B. SCSI bus
C. Controllers
D. Multiple bus

33. The main advantage of multiple bus organization over a single bus is _____.
A. Reduction in the number of cycles for execution
B. Increase in size of the registers
C. Better connectivity
D. None of the mentioned

34. Ctrl, Shift and Alt are known as _____ keys.

A. Function
B. Modifier
C. Alphanumeric
D. Adjustment

35. Which key of keyboard is used to make characters either upper or lower case?
A. ESC
B. Return
C. Shift
D. Both A and C

36. Which of the following is used in an optical mouse?
A. Infrared light
B. Light Emitting Diode (LED)
C. Sensor
D. Microwave

37. Which of the following are not input devices?
A. Webcam and Microphone
B. Bar Code Reader and Smart Card Reader
C. Optical Character Reader and Optical Mark Recognition
D. Monitor and Printer

38. Which of the following is/are input devices?
A. Track ball
B. Scanner
C. Touch screen
D. Magnetic Ink Card Reader (MICR)
E. All of the above

39. The MICR stands for _____.
A. Magnetic Ink Card Reader
B. Magnetic Ink Code Recognition
C. Meta Ink Character Recognition
D. None of these

40. The OMR stands for _____.
A. Optical Mark Recognition
B. Optical Magnetic Reader
C. Only Mark Recognition
D. Optical Markup Recognition

41. The OCR stands for _____.
A. Outsized Character Reader
B. Optical Character Reader
C. Operational Character Reader
D. Only Character Reader

42. DMA transfers data between _____.
A. Memory and processor
B. Processor and I/O devices
C. I/O devices and memory
D. All of the above

43. Which circuit implements the hardware priority interrupt unit function to determine the highest priority of simultaneously arriving various input signals ?
A. Priority Decoder
B. Priority Encoder
C. Priority Selector
D. Priority Preceeder

44. What registers are significantly incremented and decremented respectively for the transmission of each byte by Direct Memory Access (DMA)?
A. Address Register and Byte Count Register
B. Control Register and Byte Count Register
C. Transmitter Register and Byte Count Register

D. Status Register and Byte Count Register

45. What does the last instruction of each sub-routine that transfer the control to the instruction in the calling program with temporary address storage is called as?
   A. Jump to sub-routine
   B. Branch to sub-routine
   C. Return from sub-routine
   D. Call sub-routine

46. Which interface allows the cardinal provision of communicating with one particular input-output device in addition to the programming capability for operating with specific device?
   A. Parallel Peripheral Interface
   B. Serial Communication Interface
   C. Special Dedicated Interface
   D. Direct Memory Access Interface

47. Which of the following is the 1's complement of 10?
   A. 01
   B. 110
   C. 11
   D. 10

48. A computer program that converts assembly language to machine language is _____.
   A. Compiler
   B. Interpreter
   C. Assembler
   D. Comparator

49. Which access method is used for obtaining a record from a cassette tape?
   A. Direct
   B. Sequential
   C. Random
   D. All of the above

50. What is sub-routine nesting?
   A. Having multiple sub-routines in a program
   B. Using a linking nest statement to put many sub-routines under the same name
   C. Having one routine call the other
   D. None of the mentioned

Answers:

1. D	2. C	3. C	4. C	5. D
6. D	7. C	8. A	9. A	10. D
11. D	12. C	13. A	14. C	15. C
16. D	17. B	18. C	19. A	20. C
21. A	22. A	23. A	24. A	25. C
26. A	27. C	28. D	29. C	30. B
31. A	32. A	33. D	34. B	35. C
36. B	37. D	38. E	39. A	40. A
41. B	42. A	43. B	44. A	45. C
46. C	47. A	48. C	49. B	50. C

CHAPTER 6

OPERATING SYSTEM

SYLLABUS

Process, Thread, CPU Scheduling, Deadlock, Synchronization, Memory Management, Disk Management, File Management.

6.1 INTRODUCTION

The users of a system interact with the system's hardware. However, direct communication is not possible, *i.e.*, a user cannot send direct instructions to the hardware to execute various activities since it is extremely difficult for the user to transform their request or instruction into machine language that the hardware components can understand. Another issue is that the user may wish to execute several things at once, yet it will be difficult for the user to use the hardware efficiently. As a result, we will need some sort of intermediary that will allow us to efficiently access the system's physical components. So, here's where an Operating System comes in. It is a collection of software programs that forms a bridge between user instructions and hardware present within a computing device. The OS helps you to communicate with the computer without knowing how to speak the computer's language. A position of the OS within different layers of a computing system can be outline from the Fig. 6.1.

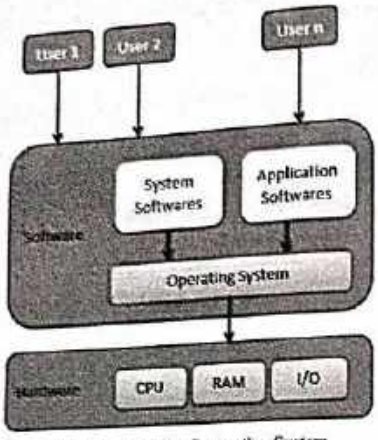


Fig. 6.1: Layers of a Computing System

6.1.1 Goal of an Operating System

The main goal of an operating system is to create a user-friendly environment that is easy to operate. The secondary purpose, on the other hand, could be described as efficiency. Efficiency in terms of resource usage in relation to time.

6.1.2 Functions of an Operating System

The important and desirable functions of an operating system are listed below:

- Process Management
- Memory Management
- File Management
- Security
- Device Management

6.2 IMPORTANT TERMINOLOGIES

6.2.1 Kernel

The kernel is the OS's primary component, forming a core that provides the OS's essential services to all other parts. A kernel is frequently contrasted with a shell, which is the most visible element of an operating system and interacts with user commands. When an operating system is loaded, kernel loads first and stays in memory until the operating system is shut off. It is in charge

of things like disc management, task management and memory management. A kernel can be of two types:

- **Monolithic Kernel:** All the operating system services run within a kernel space.
- **Micro Kernel:** Services run in different address spaces including kernel space.

6.2.2 System Call

A system call is a method for a computer application to request a service from the kernel of the operating system on which it is running. A system call is a method of interacting with the operating system via programs, *e.g.*, reading a file, creating a process, writing into device buffers, etc.

6.2.3 Batch System

In earlier computers, to speed up the processing, tasks of similar nature were grouped together for execution.

6.2.4 Spooling

Technically, "Spool" stands for "simultaneous peripheral operations online". Spooling is a method of temporarily storing data so that it can be used and executed by a device, program, or system. Data is delivered to and held in memory or other volatile storage until it is requested for execution by a programme or computer.

6.2.5 Multi-programming

Through multi-programming, maximum utilization of CPU can be made. Multiple programs can be loaded at once into main memory for execution. At any point of time, only one program or process can use the CPU to execute their instructions, while other programs must wait their turn.

6.2.6 Multi-processing

A processing system in which multiple CPUs are linked together to execute a task. Tasks are broken down into sub-tasks, which are then distributed among numerous CPUs.

6.2.7 Multi-tasking

Within a computing context, the ability to perform more than one task at the same time by a user.

6.2.8 Time-sharing System

It is based on multi-programming concepts, which allow many tasks to be handled at the same time by switching back and forth between them. Because it works without sharing the system, its switching is lightning fast, allowing users to interact with any software.

6.3 PROCESS

An operating system is basically a collection of programs required to manage the available resources such as memory, CPU, I/O devices, etc. These programs generally reside in the secondary memory however for execution a program must be loaded in the main memory. The program which is under execution by a processor is called process. Some relevant information related to a process is stored in the Process Control Block (PCB). PCB contains the following:

- **Process Id:** A unique number assigned to identify a process during process creation.
- **Process State:** A process may reside in a particular state such as new, ready, running, suspended and terminated.
- **Program Counter:** It carries the address of the next instruction to be executed by that process.
- **Register Information:** Stores the information stored in the general purpose registers of the process.
- **Accounting Information:** Amount of CPU time used for executing a process.
- **CPU Scheduling Information:** It includes address to scheduling queues, priority and etc., and list of open and closed files.

6.3.1 Process State

An operating system will create a new process. Throughout its life cycle, it goes through numerous states. A block diagram (Fig. 6.2) is represented for clear illustration:

- **New:** This state implies that the program is going to be picked by the OS into the main memory for execution.
- **Ready:** After a process is created, it enters the ready state, which means it gets loaded into the main memory. The process is ready to execute and is awaiting CPU time to complete its task. Processes that are ready to be executed by the CPU are kept in a ready process queue.
- **Running:** The CPU selects the process for execution, and the instructions within the process are executed by any of the CPU cores that are available.
- **Blocked or wait:** The process enters the blocked or wait state if it demands I/O, user input, or access to a crucial region (for which the lock has already been acquired). The process continues to wait in main memory and does not consume any CPU resources. The process returns to the ready state once the I/O operation is completed.
- **Exit (Terminated or completed):** Process is killed as well as PCB is deleted.
- **Suspend ready:** Processes that were initially in the ready state but were switched out of main memory and moved onto external storage by the scheduler are said to be in the suspend ready state. When the process is brought back onto the main memory, it will transition back to the ready state.

- **Suspend wait or suspend blocked:** Similar to suspend ready, except this time it uses the process that was executing an I/O activity and had to shift to secondary memory due to a lack of main memory. When the task is completed, it may be go to suspend ready.

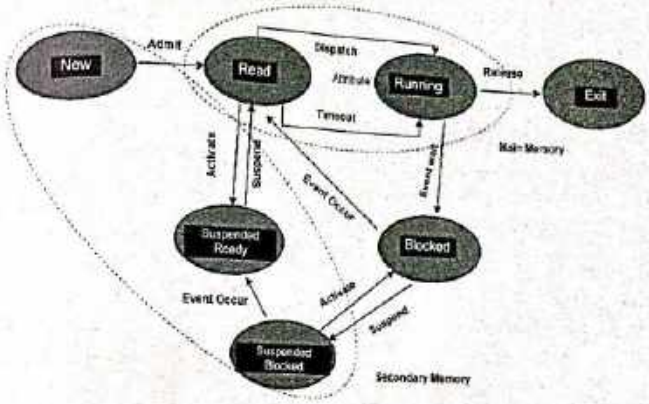


Fig. 6.2: Process State

6.4 PROCESS SCHEDULING

Process Scheduling is a task in the operating system that schedules processes in various states such as ready, waiting, and running. Process scheduling allows the operating system to assign each process a time interval for CPU execution. Another significant advantage of applying a process scheduling system is that it keeps the CPU active at all times. This allows you to get the minimum response time for programs. Scheduling falls under two categories:

- **Non-preemptive:** In non-preemptive, a resource cannot be withdrawn from a process until it has finished running. When a running process finishes and switches to a waiting state, resources are switched.
- **Preemptive scheduling:** In preemptive scheduling, the operating system assigns resources to a process for a given period of time. The process transitions from running to ready state or from waiting to ready state during resource allocation. This switching occurs because the CPU may provide priority to other processes and replace the running process with one that has a higher priority.

6.4.1 Process Scheduler

Scheduling is carried out by a scheduler. A scheduler is an operating system program (module) that selects the next job (process) to be admitted for execution. There are three different types of scheduler encountered in OS. A scheduler is activated during the following scenarios:

- When the current process changes state from running to waiting as a result of an I/O request or an undesired events.
- If the existing process comes to an end.
- When the scheduler needs to move a process from running to ready state as it has already run for its allotted interval of time.
- A process switches from the waiting state to the ready state when the requested I/O operation is done. As a result, the scheduler has the option of replacing the currently operating process with a freshly ready one.

6.4.2 Types of Scheduler

In general, three types of scheduler are present:

- Long-term scheduler
- Short-term scheduler
- Medium-term scheduler

The different schedulers are illustrated through the below comparison charts:

Table 6.1: Comparison Chart of the Schedulers

Sl. No.	Long-term scheduler	Short-term scheduler	Medium-term scheduler
1.	A job scheduler.	A CPU scheduler.	A process swapping scheduler. Removes processes from memory and makes pace for other prioritized processes.
2.	Selects a process from pool and loads it into memory for execution.	Selects a process from ready queue (ready state) that is ready for execution.	Re-introduces processes into memory for continued execution.
3.	Controls the degree of multi-programming.	Provides less control over the degree of multi-programming.	Reduces the degree of multi-programming.
4.	Slowest speed.	Fastest Speed.	Speed is between the other two.
5.	Absent or minimal in the time-sharing OS.	Minimal in time-sharing OS.	Part of time-sharing OS.

6.4.3 Context Switching

During scheduling, the scheduler shifts the CPU from one process to another. Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This is called context switching. The context of a process is represented in the Process Control Block (PCB) of a process; it includes the value of the CPU registers, the process

state and memory-management information. When a context switch occurs, the Kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run. Because the system does no valuable work while switching contexts, context transition time is pure overhead. Its speed varies by machine, depending on memory speed, the amount of registers to be duplicated, and the presence of special instructions (such as a single instruction to load or store all registers). The typical speed ranges from one to a thousand microseconds. Context switching has become such a performance barrier that programmers are resorting to new structures (threads) to prevent it as much as feasible.

The context switching is done by another program called dispatcher. It is responsible for saving and loading the context of a process.

6.4.4 Scheduling Performance Criteria

- **Arrival Time (AT):** Time at which the process arrives in the ready queue (ready state).
- **Burst Time (BT):** Amount of time required by a process to execute on CPU.
- **Completion Time (CT):** The time when a process completes its execution is called as completion time.
- **Turnaround Time (TAT):** Time interval from the time of submission (AT) to the time of completion (CT). TAT = CT – AT.
- **Waiting Time (WT):** Total time spent by a process waiting in a ready queue. WT = TAT – BT.
- **Response Time (RT):** The time interval of arrival time and first allocation of the CPU to that process.

6.4.5 Scheduling Algorithms

Some of the important algorithms are listed here:

(a) **First Come First Serve (FCFS):** It is non-preemptive scheduling. Scheduler allocates the CPU on first come first serve basis. The process which enters first in the ready queue will be allocated CPU first.

(b) **Shortest Job First Scheduling (SJF):** Among all the processes present in the ready queue, the process having least Burst Time (BT) will be the first to be allocated the CPU. In case of tie, FCFS is used. It can be both preemptive and non-preemptive.

(c) **Shortest Remaining Time First (SRTF):** Preemptive version of SJF. The process with the smallest amount of time remaining until completion is selected to execute.

(d) **Round Robin Scheduling (RR):** Basically, FCFS with time quantum (a small amount of time), *i.e.*, the CPU is allocated to the first process present in the ready queue for a time quantum. When the time quantum is over and the process still needs the CPU for execution, the process goes and waits at the end of the ready queue. Thus, ready queue behaves as a circular queue. It is also called time-sharing systems.

LRTF (Longest Remaining Time First): Preemptive

HRRN – Highest Response Ratio next : - Non preemptive (Multilevel queue)

(e) **Priority Scheduling:** Here, each process is provided with priority. The process with highest priority is assigned the CPU first. In case of tie on priority, FCFS occurs. The major drawback of this algorithm is starvation, that is, a low-priority process have to wait for an infinite time because the high priority processes gets executed first and after some time another high priority process comes and the low priority process is again left out.

6.5 PROCESS SYNCHRONIZATION

A process can be of two types:

1. **Independent Process:** A process which operates independent of other processes. The result of this process neither gets affected by any process nor affects the result of another process.
2. **Cooperating Process:** The results of this process gets affected by other processes and simultaneously, it can affect the results of other process too. These processes shares data with other processes.

At times, when several cooperating processes are executed concurrently, the final output obtained is wrong. The result leads to inconsistency of the system. This is called race condition. A race condition is a situation in which the order of execution of the threads within critical section leads to different outputs. This can be avoided using process synchronization.

6.5.1 Critical Section

A part of the program which operates on shared variables and is manipulated by the processes.

6.5.2 Critical Section Problem

A critical portion of code is a section of code that can only be accessed by one process at a time. The critical section comprises common variables that must be synced in order to ensure data consistency. Fig. 6.3 shows the various sections available in a typical computer program.

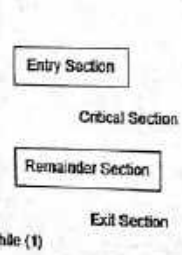


Fig. 6.3: Sections within a program

6.5.3 Solutions to Critical Section Problem

A solution to the critical section problem must satisfy these three constraints:

(a) **Mutual Exclusion:** If a process is executing in its critical section, then no other process is allowed to execute in the critical section.

(b) **Progress:** If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next, and the selection cannot be postponed indefinitely.

(c) **Bounded Waiting:** A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

6.5.4 Peterson's Solution

Peterson's Solution is a classical software based solution to the critical section problem. In Peterson's solution, we have two shared variables:

(a) **boolean flag[i]:** Initialized to FALSE, initially no one is interested in entering the critical section.

(b) **int turn:** The process whose turn is to enter the critical section.

Table 6.2: Peterson's Solution Code

P_i	P_j

do{
flag[i] = true;
turn = j;
while(flag[j] and turn = j); Critical Section
flag[i] = false
Remainder Section
}while(1);
	do{
	flag[j] = true;
	turn = i;
	while(flag[i] and turn = i); Critical Section
	flag[j] = false
	Remainder Section
	}while(1);

Peterson's Solution preserves all three conditions:

(i) Mutual Exclusion is assured as only one process can access the critical section at any time.

(ii) Progress is also assured, as a process outside the critical section does not block other processes from entering the critical section.

(iii) Bounded Waiting is preserved as every process gets a fair chance.

The disadvantages of Peterson's Solution are:

(a) It involves busy waiting.

(b) It is limited to two processes.

6.5.5 Semaphore

It is a synchronization tool that overcomes the limitations of Peterson's solution. A semaphore S is a variable that holds an integer value and is accessed only through two standard atomic operations wait() and signal(). S indicates the number of resources available in a system at a particular time.

6.5.5.1 Implementation of Semaphore

(a) **wait(S):** If the value of the semaphore variable "S" is positive, the wait() function is used to decrement the value of the semaphore variable by one. If it is 0, no operation will be executed. Originally, it was P(S) from Dutch word Proberen meaning 'to test'.

```
wait(S){
while (S == 0); // there is ";" sign here
S--;
}
```

(b) **signal(S):** The signal() function is used to increment the value of the semaphore variable by one. Originally, it was V(S) from Dutch word Verhogen meaning 'to increment'.

```
signal(S) { S++; }
```

6.5.5.2 Types of Semaphore

(a) **Binary Semaphore:** It has only two possible values: 0 and 1. Mutex locks are also known as binary semaphores because they enforce mutual exclusion.

(b) **Counting Semaphore:** The semaphore variable is first initialized with the number of resources available. The wait() method is then executed anytime a process requires a resource, and the value of the semaphore variable is decreased by one. The process then uses the resource, after which it calls the signal() function, which increases the value of the semaphore variable by one. When the value of the semaphore variable reaches 0, *i.e.*, when the process has utilized all of the resources and there are none left to be used, any other process wishing to consume resources must wait for its turn. In this way, process synchronization is achieved.

6.5.5.3 Disadvantages of Semaphore

(a) If a low priority process is in the critical section when utilizing semaphore, no other higher priority process can enter the critical section. As a result, the higher priority process must wait for the lower priority process to finish.

(b) To avoid deadlocks in the semaphore, the Wait and Signal operations are required to be executed in the correct order.

(c) With improper use, a process may block indefinitely. Such a situation is called Deadlock.

6.6 DEADLOCKS

The following lists the way how a process in an operating system uses resources:

(a) Makes a request for a resource (If available, granted immediately else need to wait)

(b) Use the available resources

(c) Resource is released after operation

A deadlock occurs when each computer process is waiting for a resource that has been allotted to another process. In this case, none of the processes can run since the resource they require is being held by another process that is also waiting for a resource to be released.

6.6.1 Necessary Conditions for Deadlocks

(a) **Mutual Exclusion:** A resource can only be shared in mutually exclusive manner. It implies, two processes cannot use the same resource at the same time.

(b) **Hold and Wait:** The process is now holding a resource and is waiting for another desired resource to become available, which is being held by other processes.

(c) **No preemption:** Resources that are already been obtained by a process cannot be returned forcibly until and unless the holding process releases them.

(d) **Circular Wait:** All processes must wait for resources in a cyclic manner, so that the last process is waiting for the resource that the first process is holding.

A deadlock can be dealt by following any of the three strategies:

(a) Deadlock prevention
(b) Deadlock avoidance
(c) Deadlock detection and recovery
(d) Deadlock ignorance

6.6.2 Deadlock Prevention

A deadlock in a system occurs when Mutual Exclusion, Hold and Wait, No Preemption, and Circular Wait are all active at the same time. If any of the four conditions can be violated at any time, the system will never have a deadlock. The concept behind the technique is simple: we must fail one of the four conditions, but its physical implementation in the system can be hard.

6.6.3 Deadlock Avoidance

It is preferable to avoid a deadlock than to take action after the deadlock has occurred. It requires futuristic information, such as how resources should be used by stating the maximum number of resources of each type that each process may require. It dynamically ensure that there never occurs a circular wait condition. To do so, three approaches present are:

1. Safe state algorithm
2. Resource allocation graph algorithm
3. Banker's algorithm

6.6.3.1 Safe State

If the system can distribute all the requested resources without causing a deadlock, then it is called a safe state. But if the system is unable to satisfy the requests of all the process present in the system, it is termed as unsafe state. A request to a resource should be granted only if it does not put the system under unsafe state.

6.6.3.2 Resource Allocation Graph Algorithm

A resource allocation graph is a graph with two types of nodes: process nodes (shown by a circle) and resource nodes (represented by rectangles). There is a dot in the resources nodes for various instances of a resource (rectangle). If there are two identical drives, for example, we will show them as two dots (one for each instance). The resource allocation and release are represented by the directed edges between these nodes. An edge from the resource to the process node indicates that the process has acquired the resource, whereas an edge from the process node to the resource node indicates that the process is making a request for the resource. Take a look at the resource allocation graph in Fig. 6.4(a).

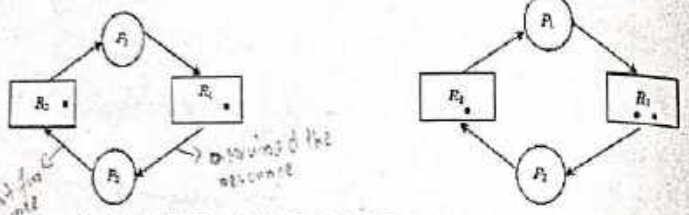


Fig. 6.4: (a) Cyclic Graph with Deadlock, (b) Cyclic but without Deadlock

The resource allocation graph $\{P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_2 \rightarrow P_1\}$ is a cyclic graph which indicates deadlock. Deadlock occurs because each resource has only one instance and each process is holding a resource while attempting for another resource which is held by another waiting process. However, Fig. 6.4(b) does have a cyclic graph but it is not deadlock.

Since the resource R_1 has two instances, the request for resource R_1 by P_1 can easily be granted as it has two instances. The points to remember are:

(a) If there is no cycle in the resource allocation graph, there is no deadlock.

(b) If there is a cycle in the graph and each resource has only one instance, then there is a deadlock. In such a case, the presence of a cycle is the necessary condition in order for deadlock to occur (as there is only one instance of the resource).

(c) If there is a cycle in the graph, and each resource has more than one instance, there may or may not be a deadlock. Therefore, we conclude that a cycle in the resource allocation graph is a necessary but not a sufficient condition for a deadlock in the case of multiple instances of a resource.

This strategy has limitations that it cannot be used in a resource allocation system with many instances of each resource type.

6.6.3.3 Banker's Algorithm

When a new process is added to the system, it must indicate how many instances of each resource type it will require. This amount should not exceed the total number of resources present in the system. For efficient execution, it requires the following data structure:

- **Available:** It keeps the data of the number of instance available of each resources type, *e.g.*, $Available[j] = k$ means resource R_j has total of k number of instances available.
- **Max:** A $n \times m$ matrix which mentions the maximum demands of each process P_i, *e.g.*, $Max[i, j] = k$ indicates process P_i requires at most k number of instances of resource R_j.
- **Allocation:** The number of resources of each type currently allotted to each process is defined by this $n \times m$ matrix, e.g., $Allocation[i, j] = k$ indicates process P_i is currently allocated k number of instances of resource type R_j.
- **Need:** This $n \times m$ matrix indicates to the number of instance of each resource type which are yet to be allocated to the process. $Need[i, j] = Max[i, j] - Allocation[i, j]$.

The Banker's algorithm consists of the following two parts:

- **Safety algorithm:** The algorithm for finding out whether a system is in a safe state can be defined as follows:

  (a) Let 'Work' and 'Finish' be vectors of lengths m and n respectively.

  $Work := Available$

  $Finish[i] := False$, for $i = 1, 2, 3, \ldots, n$

  (b) Find an 'i', such that $Finish[i] := False$, $Need_i \leq work$, if no such i exists, go to step (d).

  (c) $Work := Work + Allocation$

  $Finish[i] := true$

  Go to step (b).

  (d) If $Finish[i] := true$ for all i, then the system is in a safe state.

- **Resource Request Algorithm:** Let $Request_i$ be the request array for process P_i. $Request = [j] = k$ means process P_i wants k instances of resource type R_j. When a request for resources is made by process P_i, the following actions are taken:

  (a) If $Request <= Need_i$,

  Go to step (ii);

  otherwise, raise an error condition, since the process has exceeded its maximum claim.

  (b) If $Request_i <= Available$,

  Go to step (iii);

  otherwise, P_i must wait, since the resources are not available.

(c) Have the system pretend to have allocated the requested resources to process P_i by modifying the state as follows:

$Available = Available - Request_i$

$Allocation_i = Allocation_i + Request_i$

$Need_i = Need_i - Request_i$

6.6.4 Deadlock Detection and Recovery

- **Deadlock Detection:** Deadlock in the system can be detected easily using resource allocation graph for single instance. However, for multiple instances, detection of the cycle is necessary but not sufficient condition. In this scenario, the system may or may not be in deadlock, depending on the circumstances.
- **Deadlock Recovery:** If a deadlock is found, the operating system will utilize one of the following ways to break the conflict.

(a) **Killing the process:** One by one the processes are killed till the system is free from deadlock.

(b) **Resource Preemption:** Resources are diverted away from the processes that are stuck in a deadlock, and those resources are reassigned to other processes so that the system can be rescued. In this circumstance, the system will starve to death.

6.6.5 Deadlock Ignorance

This is the simplest of all. If the deadlock has occurred, behave as nothing has happened and restart the system.

6.7 MEMORY MANAGEMENT

Memory is a crucial component of a computer that is used to store data. Since the amount of main memory available in a computer system is fairly limited, its management is therefore becomes very important. Many processes are competing for the main memory at any given time. Furthermore, various processes run simultaneously to improve performance. To achieve degree of multi-programming, several processes must reside in the main memory. Therefore, managing them effectively and efficiently is even more crucial. The following are the main categories of memory management techniques (Fig. 6.5):

- Contiguous memory management schemes
- Non-contiguous memory management schemes

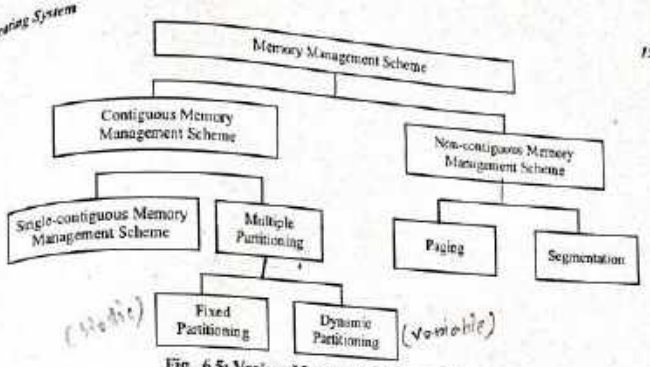


Fig. 6.5: Various Memory Management Schemes

6.7.1 Contiguous Memory Management Schemes

Each programme in a contiguous memory management scheme takes up a single contiguous block of storage locations, *i.e.*, a group of memory locations with consecutive addresses.

(a) **Single contiguous memory management scheme:** The simplest memory management method employed in the first generation of computer systems is the single contiguous memory management scheme. The main memory is separated into two contiguous sections or partitions in this approach. The operating systems are installed in one partition, which is usually in lower RAM, and the user processes are installed in the other partition.

(b) **Multiple Partitioning:** The single contiguous memory management technique is ineffective since it restricts computers to only running one programme at a time, resulting in wasting memory and CPU resources. Multi-programming, which allows multiple programmes to run at the same time, can solve the problem of inefficient CPU usage. Operating systems must load both processes into main memory in order to switch between them. To load numerous processes into main memory, the operating system must partition the available main memory into different sections. As a result, numerous processes can run in the main memory at the same time. It is of two types:

- **Fixed Partitioning:** A fixed partition memory management system divides the main memory into many fixed-size partitions. A single process can be stored in each partition. The degree of multi-programming, or the maximum number of processes in memory, is determined by the number of partitions. It suffers from internal fragmentation.
- **Dynamic Partitioning:** In this scheme, each process occupies the memory space as much as it requires. Requested processes are allocated memory until the entire

physical memory is exhausted or the remaining space is insufficient to hold the requesting process. It also suffers from internal fragmentation.

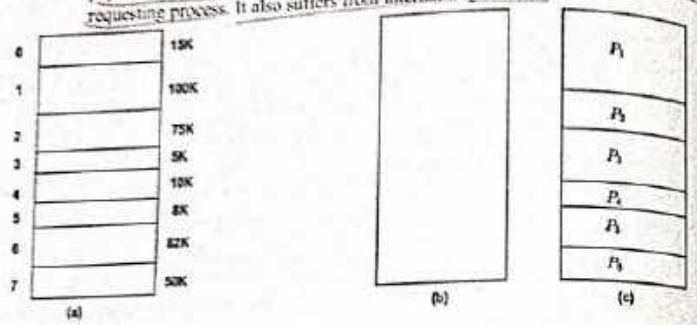


Fig. 6.6: Multiple Partitioning Schemes: (a) Fixed partition scheme, (b) Initial empty memory, and (c) After memory allocation to process

6.7.1.1 Partition Allocation Strategy

Memory allocation of the free memory spaces also called hole is assigned according to the following three strategies:

(a) **First Fit:** The process is allocated to the first free partition that meets the need of the process. The search can begin either from top or down of the memory.

(b) **Best Fit:** Assign the smallest hole that is large enough to process requirements for the optimum fit. Unless the list is organised by size, we search the entire list for this.

(c) **Worst Fit:** In the worst-case scenario, the process is assigned the largest available hole that is available in the memory. This procedure yields the greatest remaining hole.

6.7.2 Non-contiguous Memory Management Schemes

Unlike contiguous memory management scheme, a program is partitioned into different blocks and is loaded at different partitions of the memory which may not be contagious to one another. This scheme can be classified depending upon the size of blocks and whether the blocks reside in the main memory or not.

6.7.3 Fragmentation

Fragmentation is defined as a little free space (hole) created when a process is loaded and removed from memory after execution. These holes are so small that it cannot be assigned to any new process which ultimately results in memory wastage. We must limit memory waste or fragmentation in order to achieve a degree of multi-programming. There are two types of fragmentation in operating systems:

- Internal Fragmentation
- External Fragmentation

6.7.3.1 Internal Fragmentation

When a process is assigned to a memory block, a free space is created in the memory block due to the smaller size of the memory than the amount of memory requested. As a result, the empty area in the memory block is unused, resulting in internal fragmentation. For example, assume that a process P_1 of 48 MB is allocated to a memory block of 50 MB in a fixed partition. The extra memory of 2 MB would be unused and would not be allocated to any other process. This is called internal fragmentation (Fig. 6.7).

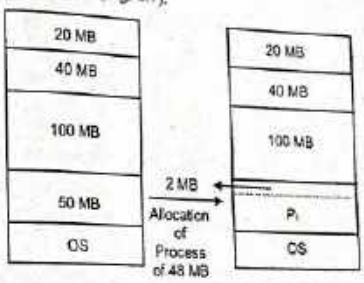


Fig. 6.7: Internal Fragmentation

6.7.3.2 External Fragmentation

When the total amount of unused space present within main memory during dynamic allocation is sufficient to satisfy a request of a process, however the memory blocks cannot be allocated since it is not contiguous. This situation is called external fragmentation. For example, assume a process P_4 of size 55 MB wants to enter the memory but the process cannot be allocated the memory blocks although the unused memory space present is 57 MB which is greater than the requested memory space. This happens due to non-availability of the unused memory space at one place (Fig. 6.8).

External fragmentation occurs since the total memory allocated to a process has to be contiguous in nature. If this condition is removed, then the external fragmentation could be avoided. This can be achieved through the concept of paging and segmentation. Compaction is another method for removing external fragmentation. External fragmentation can be reduced when dynamic partitioning is utilised for memory allocation by combining all free memory into a single large block.

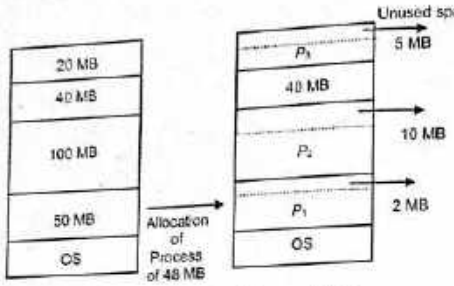


Fig. 6.8: External Fragmentation

6.7.4 Paging

This scheme permits the physical address space of a process to be non-contiguous.

(a) **Logical Address or Virtual Address (represented in bits):** An address generated by the CPU.

(b) **Logical Address Space or Virtual Address Space (represented in words or bytes):** The set of all logical addresses generated by a program.

(c) **Physical Address (represented in bits):** An address actually available on memory unit.

(d) **Physical Address Space (represented in words or bytes):** The set of all physical addresses corresponding to the logical addresses.

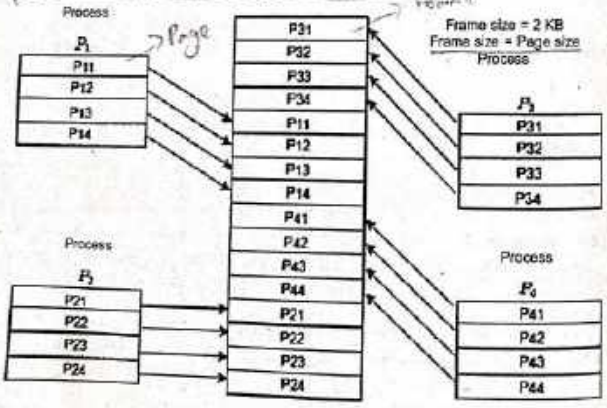


Fig. 6.9: Paging

The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames. One page of the process is to be stored in one of the frames of the memory. The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes. Pages of the process are brought into the main memory only when they are required. Otherwise, they reside in the secondary storage. Different operating system defines different frame sizes. The sizes of each frame must be equal. Considering the fact that the pages are mapped to the frames in paging, page size needs to be as same as frame size. For example, Fig. 6.9 represents the allocation of memory blocks to four processes P_1, P_2, P_3 and P_4. Size of each process is 8 KB and size of memory is 32 KB. Frame size is 2 KB.

6.7.4.1 Memory Management Unit

In order to access a page, the CPU generates the logical address. This address is mapped to a physical address by the operating system to access the page from a frame. Logical address has two parts: (i) Page Number (p) and (ii) Page Offset (d). Consider a case where physical address is represented by 12 bits, logical address by 13 bits and page size is 1K words (Fig. 6.10).

- Physical Address = 12 bits; Physical address space = $2^2 \times 2^{10} = 4K$ words ($2^{10} = 1K$)
- Logical Address = 13 bits; Logical address space = $2^3 \times 2^{10} = 8K$ words
- Number of frames = Physical address space/frame size = 4K/1K = 4; Number of bits required to represent total frame = 2
- Number of pages = Logical address space/page size = 8K/1K = 8; Number of bits required to represent total page = 3

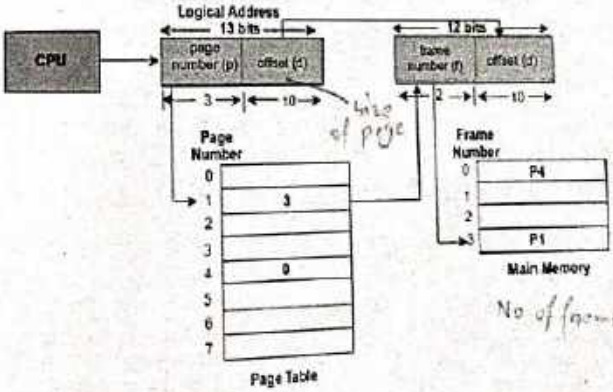


Fig. 6.10: Memory Mapping in Paging

Assume that CPU generates the logical address 001000001001. The last 3 bits represents the page number and rest 10 bits represent the offset. So, the page number that CPU wants to access is 1. To know the frame number in which the page number 1 resides, we have to look into the page table. Page table contains the frame number at which the corresponding page is loaded in the main memory. So, page number 1 is present in the frame number 3. The frame number 3 is visited to access the page number 1. Within frame number 3, offset that we are looking is 0000001001, i.e., 9th word. This word is read and transmitted to the CPU for processing.

6.7.5 Segmentation

The paging technique does not take into account the user's perspective on memory allocation. A segmentation technique is utilised to get the user's view of memory allocation (Fig. 6.11). The logical address space is partitioned into numerous segments using this technique. The size of the segments may vary. s is the segment number; it is equal to the number of bits required to represent the segment of logical address space. d is the number of bits required to represent the segment size; it is also known as segment offset. So, Number of segments in segment table = Number of segments in logical address space. Segmentation also suffers from external fragmentation.

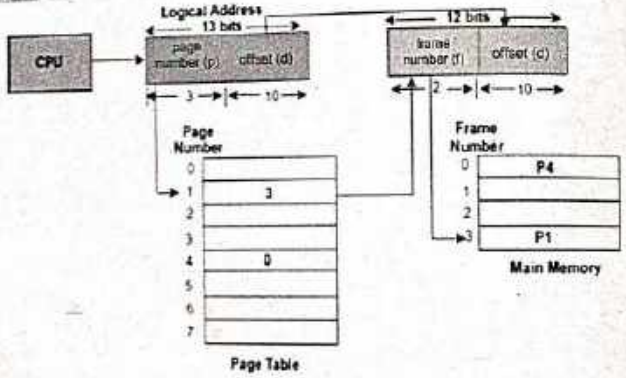


Fig. 6.11: Segmentation Mapping

6.7.6 Virtual Memory

Virtual memory is a memory management strategy that uses both hardware and software to implement it. It converts virtual addresses, which are used by programmes, into actual locations in computer memory. It gives the programmer the illusion that programmes larger than physical memory can be executed. Demand paging can be used to implement virtual memory.

6.7.6.1 Demand Paging

According to the notion of Virtual Memory, only a portion of a process needs to be present in the main memory in order for it to be executed, which means that only a few pages will be stored in the main memory at any given moment. Choosing which pages should be stored in the main memory and which should be kept in the secondary memory is a challenging task. It is hard to predicate which page will be required by a process at a specific time.

As a result, a concept known as Demand Paging was developed to address this issue. It recommends storing all of the frames' pages in secondary memory until they are needed. To put it another way, it states do not load any pages into the main memory unless they are needed. Any page that is referred to for the first time in the main memory is also found in the secondary memory.

6.7.6.2 Page Fault

Page fault takes place when the referred page by CPU is not present in the main memory. It is also called as page miss. At page fault, the specific page is looked into the secondary memory. If found, a copy of the page is stored in the main memory. When the number of page faults is high, the effective access time of the system increases dramatically. The total time taken to service a page fault is called effective access time. If the page fault rate is r%, the time spent to service the page fault is s, and the memory access time is ma, then the effective access time can be calculated as follows:

$$EAT = r * s + (1 - r) * ma$$

For example, consider a system with service time page fault = 100 ns, main memory access time = 20 ns and page fault rate = 65%. Calculate effective memory access time.

Then, $r = 65\%$, $s = 100$ ns and $ma = 20$ ns.

Applying the formula:

$EAT = 0.65 \times 100 + (1 - 0.65) \times 20 = 72$ ns.

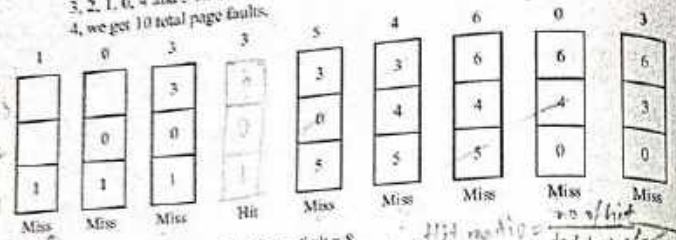
6.7.6.3 Page Replacement Algorithms

Whenever a page is requested that is not currently in memory, it must be loaded. If there isn't a free frame in memory, a page is chosen to replace it. When a page fault occurs, the page replacement algorithm determines which page should be replaced. Few algorithms are discussed:

(a) **First In, First Out (FIFO):** In this technique, a page is replaced by the page which has entered first in the memory frame. Queue data structure is used. For example, consider page reference string 1, 0, 3, 3, 5, 4, 6, 0, 3 with 3 page frames. Find number of page faults.

Belady's anomaly – Belady's anomaly shows that increasing the number of page frames while employing the First in First Out (FIFO) page replacement mechanism can result in additional page faults. When we evaluate the reference string 3, 2, 1, 0, 3, 2, 4,

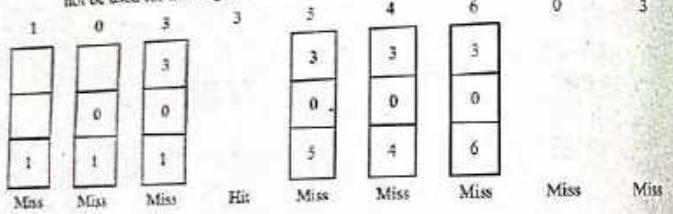
3, 2, 1, 0, 4 and 3 slots, we obtain 9 total page faults. However, when we raise slots to 4, we get 10 total page faults.



Total page fault = 8

Fig. 6.12: First In, First Out Page Replacement Algorithm

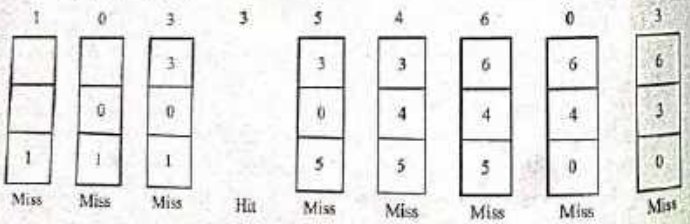
(b) Optimal Replacement Algorithm: In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.



Total page fault = 6

Fig. 6.13: Optimal Replacement Algorithm

(c) Least Recently Used: In this algorithm, page will be replaced which is least recently used.



Total page fault = 8

Fig. 6.14: Least Recently Used Page Replacement Algorithm

6.7.7 Thrashing

When a process spends more time paging or swapping than executing, it is said to be thrashing (Fig. 6.15). This leads to the decrease in throughput of the system. When the CPU utilisation is low, the process scheduling mechanism loads multiple processes into memory at the same time to improve the Degree of Multi-programming. In this case, the number of processes in memory exceeds the number of frames accessible. Each process is given a set number of frames to work with. If a higher priority process arrives in memory and the frame is not vacant at the moment, the other process that occupied the frame will be moved to secondary storage, and the free frame will be allotted to a newly arrived higher priority process. In other words, when the memory fills full, the process begins to spend a significant amount of time waiting for the required pages to be swapped in. As a result, CPU usage drops as most processes wait for pages.

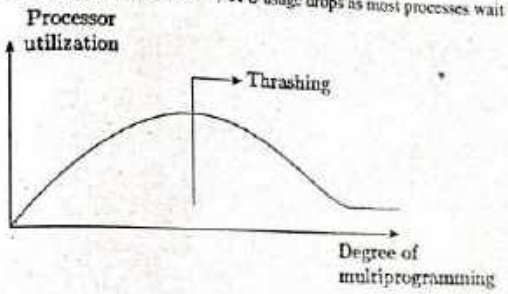


Fig. 6.15: Thrashing in OS

6.8 DISK MANAGEMENT

The difference in speed of CPU and access time of disk motivates to manage disk with efficient algorithms. A layout of the disk structure is shown in Fig. 6.16.

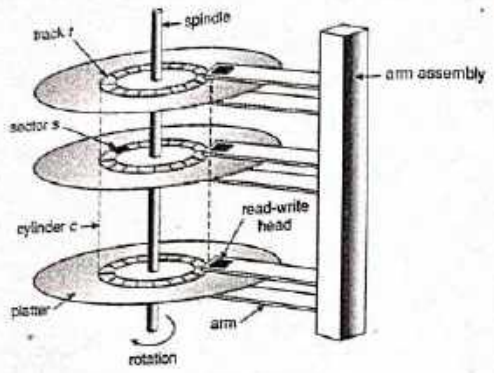


Fig. 6.16: Disk Structure

6.8.1 Time Terminology

(a) **Seek Time:** A disk is divided into many circular tracks. Seek time is the amount of time required to move the read/write head from its current position to desired track.

(b) **Rotational Latency:** Tracks in disk are further divided into blocks called sectors. Rotational Latency is the time it takes for the read/write head to rotate to the requested sector from its current position.

(c) **Transfer Time:** The amount of time taken to transfer the required data is called as transfer time. Transfer time depends on the total size of the track and the rotational rate of the disk.

(d) **Transfer Rate:** The number of bytes transferred in one unit time is called as the transfer rate of the disk.

(e) **Disk Access Time:** Disk Access Time = Seek Time + Rotational Latency + Transfer Time.

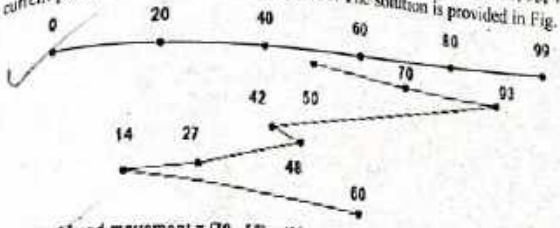
6.8.2 Disk Scheduling Algorithms

A role of an operating system is to efficiently use the hardware present in the system. Proper use of algorithm may help in fast disk access time. Some important disk scheduling algorithms are:

6.8.2.1 FCFS Scheduling Algorithm

Request is served on first come first basis. It results in no starvation and no indefinite delay. The disadvantage is that the disk scheduling time is not optimized. For example, consider a disk

contains 100 tracks (0-99) and the request queue contains track no. 70, 93, 42, 48, 27, 14, 60. The current position of the read/write head is 50. The solution is provided in Fig. 6.17.



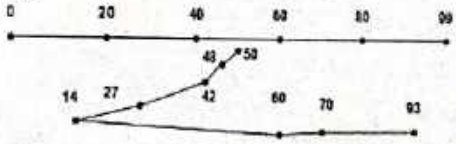
Total head movement = (70 - 50) + (93 - 70) + (93 - 42) + (48 - 42) + (48 - 27) + (27 - 14) + (60 - 14)

= 20 + 23 + 51 + 6 + 21 + 13 + 46 = 180

Fig. 6.17: FCFS Scheduling Algorithm

6.8.2.2 SSTF Scheduling Algorithm

It fulfills the request with the shortest seek time from the present head position. It is better than FCFS but causes starvation. For example, consider a disk contains 100 tracks (0-99) and the request queue contains track no. 70, 93, 42, 48, 27, 14, 60. The current position of the read/write head is 50. In order to solve this, find the track to which the read/write will move in least time. If the head moves to 70 from present position 50, the seek time is 20 (70 – 50). In similar way, the head has seek time of 43, 8, 2, 23, 36 and 10 for the track 93, 42, 48, 27, 14 and 60 respectively. Hence, it will choose to move to 48 because it has the least seek time. The complete solution is provided in Fig. 6.18.



Total head movement = (50 - 48) + (48 - 42) + (42 - 27) + (27 - 14) + (60 - 14) + (70 - 60) + (93 - 70)

= 2 + 6 + 15 + 13 + 46 + 10 + 23 = 115

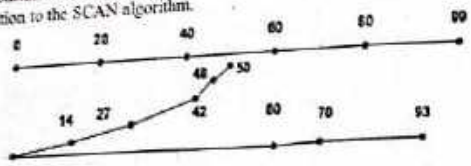
Fig. 6.18: SSTF Scheduling Algorithm

6.8.2.3 SCAN Scheduling Algorithm

Elevator Algorithm is another name for it. In this algorithm, the disc arm advances in one way until it reaches the end, satisfying all requests in its path, before turning around and moving in the opposite direction, satisfying requests in its path. It works in the same way that an elevator

does. It advances in one direction until it reaches the last floor of that direction, then reverses direction.

Consider the same example give above with initial left direction movement. Fig. 6.19 shows the solution to the SCAN algorithm.



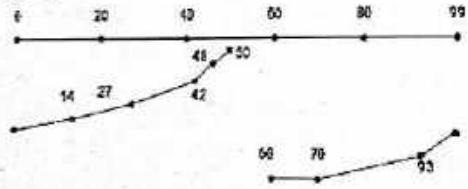
Total head movement = (50 - 48) + (48 - 42) + (42 - 27) + (27 - 14) + (14 - 0) + (60 - 0) + (70 - 60) + (93 - 70)

= 2 + 6 + 15 + 13 + 14 + 60 + 10 + 23 = 143

Fig. 6.19: SCAN Scheduling Algorithm

6.8.2.4 C-SCAN Scheduling Algorithm

In the C-SCAN algorithm, the disk's arm advances in one direction, servicing requests until it reaches the last cylinder, then jumps to the final cylinder in the opposite direction, without processing any requests, before turning back and servicing the remaining demands. An example is shown in Fig. 6.20.



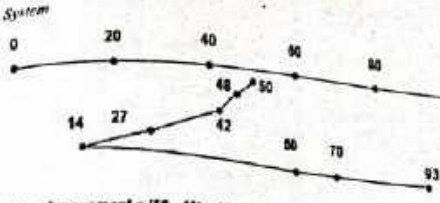
Total head movement = (50 - 48) + (48 - 42) + (42 - 27) + (27 - 14) + (14 - 0) + (99 - 0) + (99 - 93) + (93 - 70) + (70 - 60)

= 2 + 6 + 15 + 13 + 14 + 99 + 6 + 23 + 10 = 188

Fig. 6.20: C-SCAN Scheduling Algorithm

6.8.2.5 LOOK Scheduling Algorithm

To some extent, it is similar to the SCAN scheduling technique, with the exception that in this scheduling algorithm, the disc arm stops advancing inwards (or outwards) when there are no more requests in that direction. A solved example is shown in Fig. 6.21.



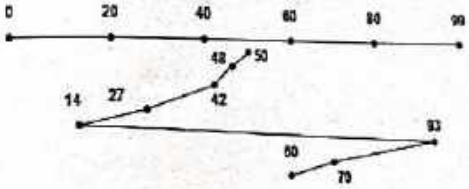
Total head movement = (50 - 48) + (48 - 42) + (42 - 27) + (27 - 14) + (60 - 14) + (70 - 60) + (93 - 70)

= 2 + 6 + 15 + 13 + 46 + 10 + 23 = 115

Fig. 6.21: LOOK Scheduling Algorithm

6.8.2.6 C-LOOK Scheduling Algorithm

The C-LOOK disc scheduling algorithm is an improved variant of both the SCAN and LOOK disc scheduling methods. In this algorithm, the head only services requests in one direction (either left or right) until all requests in that direction are not serviced, then jumps back to the farthest request in the opposite direction and services the remaining requests, resulting in better uniform servicing and avoiding wasting seek time by going to the end of the disc. A solved example is shown in Fig. 6.22.



Total head movement = (50 - 48) + (48 - 42) + (42 - 27) + (27 - 14) + (93 - 14) + (93 - 70) + (70 - 60)

= 2 + 6 + 15 + 13 + 79 + 23 + 10 = 148

Fig. 6.22: C-LOOK Scheduling Algorithm

6.9 SOLVED QUESTIONS

1. Which of the following types of operating system is non-interactive?
   A. Multi-tasking operating system
   B. Batch processing operating system
   C. Multi-user operating system
   D. Multi-programming operating system

Solution: **(B).** Batch processing operating system does not interact directly with computer. Similar jobs are grouped together into batches.

2. Time taken to switch between user and kernel modes is _____ the time taken to switch between two processes.
   A. More than
   B. Less than
   C. Independent of
   D. Equal to

   Solution: (B). Only a single mode bit has to be changed at hardware level to toggle between user and kernel mode. On the other hand, switching between processes is time-consuming due to saving of data of PCB as well as registers.

3. It is the deadlock condition in which at least one resource must be held in a non-sharable mode. Here, 'It' refers to _____.
   A. No pre-emption
   B. Circular wait
   C. Mutual exclusion
   D. Hold and wait

   Solution: (C). It refers to the Mutual Exclusion condition which leads to deadlock situation in a system. In this, only one process at a time can use the resource.

4. Which of the following requirements satisfy a solution to the critical section problem?
   A. Circular waiting
   B. Bounded waiting
   C. Unbounded waiting
   D. None of these

   Solution: (B). A solution to the critical section problem must satisfy the following three requirements – Mutual Exclusion, Progress and Bounded

waiting. Bounded waiting provides a limit on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

5. Which strategy suffer from external fragmentation?
   A. Best fit
   B. First fit
   C. First fit and best fit
   D. None of the above

   Solution: (C): Both first fit and best fit strategies for memory allocation suffer from external fragmentation. External fragmentation exists when there is enough total memory space to satisfy a request but the available spaces are not contiguous.

6. Which memory management scheme permits the physical address space of a process to be non-contiguous?
   A. Segmentation
   B. Fragmentation
   C. Paging
   D. None of these

   Solution: (C). This is the main advantage of paging over memory segmentation that it allows the physical address space of a process to be non contiguous.

7. Sub-states in a parallel program are often referred to as _____.
   A. Threads
   B. Fibres
   C. Processes
   D. None of the above

   Solution: (A). A thread of execution is the smallest sequence of programmed instructions that can be managed independently by an operating scheduler. It is a light-weight process.

8. In a paged memory, the page hit ratio is 0.35. The time required to access a page in secondary memory is equal to 100 ns. The time required to access a page in primary memory is 10 ns. The average time required to access a page is _____.
   A. 3.0 ns
   B. 68.0 ns
   C. 68.5 ns
   D. 78.5 ns

   Solution: (C). Hit ratio = 0.35, Miss ratio = 1 – 0.35 = 0.65. So, Average time = 35 * 10 + 0.65 * 100 = 68.5 ns.

9. To avoid the race condition, the number of processes that may be simultaneously inside their critical section is _____.
   A. 8
   B. 0
   C. 16
   D. 1

   Solution: (D): Race condition brings in the idea of critical section. Critical section is a mutually exclusive section whereby more than 1 process reading or writing the shared data is prohibited.

10. _____ can be used to control access to a given resource.
    A. Counting semaphores
    B. Control semaphores
    C. Concurrent semaphores
    D. None of these

   Solution: (A). Counting semaphores can be used to control access to a given resource consisting of a finite number of instances. The semaphore is initialized to the number of resources available.

11. Which of the following is a classic problem of synchronization?
    A. The Bounded problem
    B. The Dinning philosopher problem
    C. The Readers problem
    D. None of these

    Solution: (B): Classic problems of synchronization are: (i) The Dinning philosopher problem, (ii) The Readers-Writers problem, (iii) The Bounded Buffer problem.

12. If we consider a memory using a page size of 4 bytes and physical memory of 16 bytes, then what will be the number of frames?
    A. 16
    B. 4
    C. 0
    D. None

    Solution: (B). Size of page = 4 bytes, size of physical memory = 16 bytes.

    Then, Number of frames = Size of physical memory/size of page = 16 / 4 = 4 frames.

13. Which scheduling among the following allow a process to move between queues?
    A. Multi-level queue scheduling
    B. Multi-level feedback queue scheduling
    C. Round Robin scheduling
    D. Shortest job first scheduling

Solution: (B): In multi-level feedback queue scheduling, if the process uses too much time, it will be moved to a lower priority queue.

14. Which scheduling is designed for time sharing system?

A. Round Robin scheduling

B. Priority scheduling

C. Shortest job first scheduling

D. FCFS scheduling

Solution: **(A).** Round Robin scheduling is specifically designed for the time sharing system because in this a small unit of time called time quantum or time slice is defined.

15. CPU scheduling decisions take place under one of four conditions, except _____.

A. Process switches from the running state to waiting state

B. Process switches from the running state to ready state

C. Process switches from waiting state to ready state

D. A process do not terminate

Solution: **(D).** A CPU scheduling occurs when a process terminates.

16. Which of the following system is characterized by increased system throughput?

A. Uniprocessor system

B. Multi-programming system

C. Time sharing system

D. Multi-processing system

Solution: **(C).** A multi-processing system includes more than one independent processing unit. It is characterized by increased system throughput and application speedup parallel processing.

17. A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units, then _____.

A. Deadlock may occur

B. Deadlock has to occur

C. Deadlock can never occur

D. None of these

Solution: **(C).** Distribute each process to one less than maximum request resources, *i.e.*, $3 \times (2-1) = 3$.

Total number of resources = 3 + 1 (additional resource for deadlock avoidance)

If total number of resources is greater than given resources, then a deadlock will occur, *i.e.*, $3 + 1 \leq 4$. Hence, deadlock can never occur.

18. On a disk with 1000 cylinders (0 to 999), find the number of tracks, the disk arm must move to satisfy all the requests in the disk queue. Assume the last request service was at track 345 and the head is moving towards track 0. The queue in FIFO order contain requests for the following tracks: 123, 874, 692, 475, 105, 376. Assume SCAN algorithm.

A. 2013

B. 1219

C. 1967

D. 1507

Solution: **(B).** Given queue = 123, 874, 692, 475, 105, 376.

Head starts at 345 and move towards 0. So, disk arms will start from 345 and serve the nearby request in the direction of 0. So, the following head movements will occur: 345 – 123 = 222; 123 – 105 = 18; 105 – 0 = 105; 376 – 0 = 376; 475 – 376 = 99; 692 – 475 = 217; 874 – 692 = 182. Total movements = 222 + 18 + 105 + 376 + 99 + 217 + 182 = 1219.

19. Consider the following program:

```
main()
{
  if(fork()>0)
sleep(100);
}
```

A. Infinite process

B. Orphan process

C. Zombie process

D. None

Solution: **(A).** zombie process or defunct process is a process that has completed execution (via the exit system call) but still has an entry in the process table: it is a process in the "Terminated state"

20. Consider the following set of processes with the length of the CPU burst time given in milliseconds.

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

What will be the average waiting time of shortest job first scheduling?

A. 8 ms

B. 7 ms

C. 17 ms

D. None of these

Solution: **(B).** First, form the Gantt chart as given below:

Waiting time of $P_1 = 3$ ms:

P_4	P_1	P_3	P_2
0	3	9	16	24

Waiting time of $P_2 = 16$ ms; Waiting time of $P_3 = 9$ ms; Waiting time of $P_4 = 0$ ms. So, average waiting time = $(3 + 16 + 9 + 0)/4 = 7$ ms.

21. Consider the following set of jobs with their arrival times, execution time (in ms) and deadlines.

Calculate the throughput for FCFS scheduling algorithm.

Job	Arrival	Execution Time	Deadline Time
1	0	5	5
2	1	15	25
3	3	12	10
4	7	25	50
5	10	5	12

A. 0.21

B. 0.23

C. 0.1

D. 0.081

Solution: **(D).** Turnaround time = Terminated time – Arrival time. So, turnaround times for various jobs are: Job 1: 5 – 0 = 5 ms; Job 2: 20 – 1 = 19 ms; Job 3: 32 – 3 = 29 ms; Job 4: 57 – 7 = 50 ms; Job 5: 62 – 10 = 52 ms.

Throughput = Number of process completed per unit time = 5/62 = 0.081.

22. Page replacement policy should satisfy _____.

A. Storage capacity

B. Page fault rate must increase

C. Non-interference

D. None of these

Solution: (C). The following are the main requirements that should be met by a page replacement policy: (i) the page replacement strategy must not interfere with the program's locality. (ii) it must not remove a page that may be referenced in the near future. and (iii) the page fault rate must not grow as the memory allocation for a program increases.

23. A process executes the code:

fork ();

fork ();

fork ();

A. 3

B. 4

C. 7

D. 8

Solution: (C). fork() system call is used to create child processes. Total number of processes: 2n. Child processes: 2n − 1. Here, n represents the number of fork calls.

4. Which of the following is NOT true of deadlock prevention and deadlock avoidance schemes?

A. In deadlock prevention, the request for resources is always granted if the resulting state is safe.

B. In deadlock avoidance, the request for resources is always granted if the result state is safe.

C. Deadlock avoidance is less restrictive than deadlock prevention.

D. Deadlock avoidance requires knowledge of resource requirements a priori.

Solution: (C). Deadlock avoidance scheme is less restrictive than deadlock prevention because in the latter, the request for a resource may not be granted even if the resulting state is safe.

25. Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are, respectively,

A. 256 Mbyte, 19 bits

B. 256 Mbyte, 28 bits

C. 512 Mbyte, 20 bits

D. 64 Gbyte, 28 bits

Solution: (A). Disk capacity = 16 surfaces × 128 tracks × 256 sectors × 512 bytes = 256 MB. Total number of sectors = 16 × 128 × 256 = 219.

Answers:

1. B	2. B	3. C	4. B	5. C
6. C	7. A	8. C	9. D	10. A
11. B	12. B	13. B	14. A	15. D
16. C	17. C	18. B	19. A	20. B
21. D	22. C	23. C	24. C	25. A

6.10 PRACTICE QUESTION SETS

1. One can interface with operating system by means of _____.

A. Operating system call in a program

B. Operating system commands

C. Operating system process

D. Both by operating system call and operating system commands

2. Kernel is _____.

A. A part of operating system

B. An operating system

C. A hardware

D. A register

3. Virtual machine are _____.

A. Lack of machine

B. Illusion of real machine

C. Physical machine

D. None of these

4. TLB is the acronym of _____.

A. Translation Looking Buffer

B. Translation Left Buffer

C. Translation Look aside Buffer

D. Translation Look Block

5. Replacement policies for different pages is required when _____.

A. There is no unused page frames in main memory

B. Allocation of new incoming item

C. Both A and B

D. None of these

6. Which of the following scheduler is in charge of handling the swapped out process?

A. Long term

B. Medium term

C. Short term

D. None of these

7. Only the process executing the critical section is allowed access to the shared variable, all other processes should be prevented from doing so until the completion of the critical section. This is often referred to as _____.

A. Mutual Exclusion

B. Semaphores

C. Deadlock

D. Inter-process communication

8. It is the deadlock condition in which at least one resource must be held in a non-sharable mode. Here, 'It' refers to _____.

A. No pre-emption

B. Circular wait

C. Mutual exclusion

D. Hold and wait

9. Which of the following requirements satisfy a solution to the critical section problem?

A. Circular waiting

B. Bounded waiting

C. Unbounded waiting

D. None of these

10. Access time is _____.

A. Seek time + Latency time

B. Seek time

C. Seek time − Latency time

D. Latency time

11. _____ is a technique of temporarily removing inactive programs from the memory of computer system.

A. Swapping

B. Spooling

C. Semaphore

D. Scheduler

12. Consider a logical address space of 8 pages of 1024 words mapped with

memory of 32 frames. How many bits are there in the physical address?

A. 9 bits
B. 11 bits
C. 13 bits
D. 15 bits

13. Program 'pre-emption' is _____.
A. release of CPU by the program after completing its task
B. forced deallocation of the CPU from a program which is executing on the CPU
C. a program terminating itself due to detection of an error
D. forced allotment of CPU by a program to itself

14. Which of the following is NOT included in the process?
A. Program counter
B. Stack
C. Data section
D. Increment counter

15. In which of the following algorithm operating system swaps out the page whose next use will occur farthest in future?
A. NRU algorithm
B. LIFO algorithm
C. FIFO algorithm
D. Optimal page replacement algorithm

16. Find odd one out:
A. A blocking kernel-scheduled thread blocks all threads in the process
B. Kernel-scheduled threads are cheaper to create than user-level threads
C. All kernel-scheduled threads of a process share the same virtual address space
D. System calls do not change the privilege mode of the processor

17. Jobs which are admitted to the system for processing is called _____.
A. Short-term scheduling
B. Long-term scheduling
C. Medium-term scheduling
D. Queuing

18. In which of the following page replacement policies Belady's anomaly occurs?
A. FIFO
B. LRU
C. LFU
D. NRU

19. The _____ strategy produces smallest leftover hole.
A. Best-fit
B. First-fit
C. Last-fit
D. FIFO

20. Page fault frequency in an operating system is reduced when the _____.
A. Processes tend to the I/O bound
B. Size of pages is reduced
C. Processes tend to be CPU-bound
D. Locality of reference is applicable to the process

21. Consider the virtual page reference string:

1, 2, 3, 2, 4, 1, 3, 2, 4, 1

on a demand paged virtual memory system running on a computer system with main memory size of 3 pages frames which are initially empty. Let LRU, FIFO and OPTIMAL denote the number of page faults under the corresponding page replacements policy. Then _____.
A. OPTIMAL < LRU < FIFO
B. OPTIMAL < FIFO < LRU
C. OPTIMAL = LRU
D. OPTIMAL = FIFO

22. Process no longer exist, but it leaves a record for its parent process to collect, the state of the process is _____.
A. Spawned
B. Slipend
C. Zombie
D. Running

23. Consider a system having m resources of the same type. These resources are shared by three processes A, B and C which have peak demands of 3, 5 and 7, respectively. For what value of m, deadlock will not occur?
A. 13
B. 14
C. 15
D. 16

24. Context switching is part of _____.
A. Buffering
B. Process execution state
C. Interrupt handling
D. Ready state

25. The time taken to move the desired sector under the RW head is called _____.
A. Seek time delay
B. Rotational time delay
C. Transfer time delay
D. Queuing time delay

26. The average page fault service time in a system is 25 ms and memory access time is 220 ns. If page fault rate is 0.0004, what will be the effective access time?
A. 10
B. 7.36
C. 10.21
D. 11.36

27. The capacity of memory unit is defined by the number of words multiplied by the number of bits/word. How many separate address and data lines are needed for a memory of 4K × 16?
A. 10 address, 16 data lines
B. 11 address, 8 data lines
C. 12 address, 16 data lines
D. 12 address, 12 data lines

28. What is the number of page faults by least recently used page replacements for a memory with 4 frames for the page reference string 2, 0, 1, 2, 4, 0, 5, 1, 4, 6, 4, 2, 1, 3, 0?
A. 7
B. 8
C. 9
D. 10

29. What is the ready state of a process?
A. When process is scheduled to run after execution

B. When process is unable to run until some task has been completed

C. When process is using the CPU

D. None of the mentioned

30. Which is not the function of the Operating System?

A. Memory management

B. Disk management

C. Application management

D. Virus protection

31. A multi-processing operating system cannot be implemented on hardware that does not support _____.

A. Address translation

B. DAM for disk transfer

C. Demand paging

D. All of the above

32. Thrashing occurs in a system when _____.

A. The processes on the system access pages and not memory frequently

B. A page fault pops up

C. The processes on the system are in running state

D. The processes on the system are in the waiting state

33. Which one of these is NOT shared by the same process's threads?

A. Address Space

B. Stack

C. Message Queue

D. File Descriptor Table

34. Out of these statements, which one are/is true?

P. The shortest remaining time in the first scheduling may lead to starvation

Q. Pre-emptive scheduling may lead to starvation

R. In terms of responsive time, Round Robin is comparatively much better than FCFS

A. P only

B. P and R only

C. Q and R only

D. P, Q and R

35. An OS utilizes the SRT or Shortest Remaining Time First Process Scheduling Algorithm. Let us consider the execution time and arrival time for these processes: The total waiting time for the

Job	Arrival Time	Execution Time
P_1	0	20
P_2	15	25
P_3	30	10
P_4	45	15

P_2 process would be _____.

A. 55

B. 40

C. 15

D. 5

36. We can define a thread as a "lightweight process". It is because an OS (operating system) maintains much shorter data structures for a thread instead of a process. Concerning this, which of the following statements is TRUE?

A. The Operating System maintains only the CPU register state on a per-thread basis

B. The OS does not maintain each thread's separate stack

C. The Operating System does not maintain a virtual memory state on a per-thread basis

D. The Operating System maintains only accounting and scheduling information on a per-thread basis

37. An OS implements a policy that needs a process to release all of the resources before it makes any requests for another resource. Out of all the statements below, select the one that is TRUE:

A. Both deadlock and starvation can occur

B. Deadlock cannot occur, but starvation can occur

C. Deadlock can occur, but starvation cannot occur

D. Neither deadlock nor starvation can occur

38. A solution to the Dining Philosophers Problem which avoids deadlock is _____.

A. Ensure that all philosophers pick up the left fork before the right fork

B. Ensure that all philosophers pick up the right fork before the left fork

C. Ensure that one particular philosopher picks up the left fork before the right fork, and that all other philosophers pick up the right fork before the left fork

D. None of these

39. How an operating system does make sure that a particular program has had a successful completion?

A. The return value is 0.

B. The return value is 1.

C. The return value is –1.

D. The program does not return a value.

40. Increasing the RAM of a computer typically improves performance because _____.

A. Virtual memory increases.

B. Larger RAMs are faster.

C. Fewer page faults occur.

D. Fewer segmentation faults occur.

41. Which of the following is an example of SPOOLED device?

A. The terminal used to enter the input data for a program being executed

B. The secondary memory device in a virtual memory system

C. A line printer used to print the output of a number of jobs

D. None of the above

42. Critical region is _____.

A. A part of the operating system which is not allowed to be accessed by any process

B. The software which monitors the operating system

C. The set of primitive functions upon which the rest of operating system functions are built up

D. None of the above

43. The size of the virtual memory depends on _____.

A. The size of the data bus
B. The size of the main memory
C. The size of the address bus
D. None of the above

44. Suppose that a process is in 'BLOCKED' state waiting for some I/O service. When the service is completed, it goes to the _____.
A. Running state
B. Ready state
C. Suspended state
D. Terminated state

45. The main disadvantage of semaphores is that _____.
A. They require large amounts of memory
B. They are very hard to program
C. They consume processor time
D. They are an incomplete solution

46. The first program that a computer runs when it is powered up or rebooted is a _____.
A. Application program
B. Monitor program
C. Kernel
D. Bootstrap program

47. Which of the following is a service not supported by the operating system?
A. Protection
B. Accounting
C. I/O operation
D. Compilation

48. Overlay is _____.
A. A part of an operating system
B. A single memory location
C. A single contiguous memory that was used in olden days for running large programs by swapping
D. Overloading the system with many user files

49. Which of the following scheduling algorithm is non-preemptive?
A. FCFS scheduling
B. SJF scheduling
C. Priority scheduling
D. Round Robin scheduling

50. One of the problems with priority scheduling is _____.
A. Aging
B. Starvation
C. Process death
D. Average waiting time

Answers:

1. D	2. A	3. B	4. C	5. C
6. B	7. A	8. C	9. B	10. A
11. A	12. D	13. B	14. D	15. D
16. C	17. B	18. A	19. A	20. D
21. B	22. C	23. D	24. C	25. B
26. C	27. C	28. D	29. A	30. D
31. A	32. A	33. B	34. D	35. C
36. C	37. B	38. A	39. A	40. C
41. C	42. B	43. C	44. B	45. C
46. D	47. D	48. C	49. A	50. B

❐ ❐ ❐

CHAPTER 7
COMPUTER NETWORK

SYLLABUS

Concepts of Networking, Application Areas, Classification, Reference Models, Transmission Environment and Technologies, Routing Algorithms, IP, UDP and TCP Protocols, IPv4 and IPv6, Reliable Data Transferring Methods, Application Protocols, Network Security, Management Systems, Perspectives of Communication Networks.

7.1 INTRODUCTION

Computer networking is the process of electronically linking two or more computing devices to exchange information through data connections. It can also include multiple devices/ mediums which help in the communication between two different devices. These are known as network devices and include things such as routers, switches, hubs, and bridges. Although the security requirements may be different depending upon the application, network security cannot be left aside when studying networks.

7.2 CONCEPT OF NETWORKING

Two or more devices connecting to each other through any medium forms a network. To connect the devices, there are two possible ways:

- **Point-to-point connection:** This provides a dedicated link between the two devices.
- **Multi-point connection:** In this case, channel capacity is shared by more than two devices. This is also known as **multi-drop** connection.

7.2.1 Network Topology

Topology defines the structure of the network of how all the components are interconnected to each other. The various network topologies are given in Fig. 7.1:

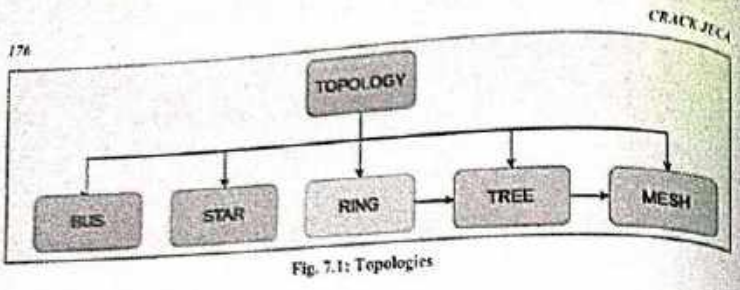


Fig. 7.1: Topologies

7.2.1.1 Bus Topology

In a bus network, all the devices are connected with one cable. The major benefit here is, we require less cable length than star topology and expansion is quite easy with the help of repeaters. The issues associated with bus topology are as follows:

- Only one device can send the data at one time. All the devices will listen at that time.
- The data communication is only in one direction.
- In a bus topology, if the network shuts down then there is problem in identifying the culprit device.

7.2.1.2 Ring Topology

In a ring topology, each device is connected exactly to two devices to form the ring. Repeaters are used to regenerate and retransmit each bit. Data travels around the network in one direction. Data travels in the form of token. Additional components do not affect the performance of the network. Even if the load on the network increases, the performance of a ring topology is better than a bus topology. The problems may be listed as follows:

- Failure of one computer in the ring may lead to entire communication loss.
- Network scaling is difficult

For example, token ring is defined by IEEE 802.5 standard.

7.2.1.3 Star Topology

In a star topology, a hub is placed at the central location and all the devices are connected to the hub. All communication is possible through the hub only. If the hub is active, it may amplify or regenerate the signals. The following are the drawbacks of a star topology:

- If the central hub fails, no communication is possible.
- Cabling cost is more.

For example, Ethernet 10Base-T is a popular example.

7.2.1.4 Mesh Topology

All the devices are connected through peer-to-peer links. A fully connected mesh will have $n(n-1)/2$ physical channels to connect n devices. The advantage of mesh topology is security and privacy. A dedicated link will eliminate traffic problems, and fault diagnose is easy here. But cabling cost and other hardware required make it difficult to implement in real practice. It is better to use this topology in backbone network and other topologies for further network configuration.

7.2.1.5 Tree Topology

A tree topology maintains the devices connected to a central hub as well as to some secondary hubs, which are again connected to the central hub. It allows an isolated network which prioritizes communication from different computers. It also faces the same problem as in a star topology; failure of central hub will crash the entire network. Also, it has high cabling cost.

Ring and mesh topology are examples of peer-to-peer relationship because here all the devices share the link equally. Bus, star and tree topologies are examples of primary-secondary relationship, where one device controls and the other devices have to transmit through it.

7.3 LAN TECHNOLOGIES

LAN (local area network) is an integral part to create a network. There are many LAN technologies such as Ethernet, token ring, token bus, FDDI (fiber distributed data interface) and ATM LAN.

7.4 ETHERNET

Xerox Corporation, Digital Equipment Corporation and Intel Corporation developed Ethernet LAN technology in 1976. Ethernet is based on the IEEE 802.3 specification. It is a linear-bus logical topology. Ethernet is the most widely used LAN technology in the world. It has passed four generations: Standard Ethernet (10 Mbps), Fast Ethernet (100 Mbps), Gigabit Ethernet (1 Gbps) and Ten-Gigabit Ethernet (10 Gbps). Earlier, Ethernet designed were of two types: Thicknet (thick coaxial main trunk cable of 10 mm and Thinnet (thin coaxial cable: RG-58 of 5 mm). In 1990, unshielded twisted-pair (10Base-T) Ethernet came into existence. Ethernet uses Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access control scheme. The drawback of this topology is that if one of the links between the two adjacent nodes fails, the whole network fails.

7.4.1 Token Bus

Token bus is a bus (physical view) ring (logical view) topology. In token bus, nodes are connected linearly. However, they make a logical ring, as each node knows the address of its successor.

7.4.2 Token Ring

Token ring is a star (physical view) ring (logical view) topology. The hub acts as a connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. It is more efficient, if a link goes down, it will bypass the hub and operate the other nodes. It also improves the scalability of the network.

In early token release,

$$\text{Throughput for single station or N stations} = \frac{D}{T_t + (R_t + N_t)}$$

In delayed token release,

$$\text{Throughput for single station} = \frac{D}{T_t + R_t + (R_t / N_t)}$$

$$\text{Throughput for N stations} = \frac{D}{R_t + (R_t / N_t)}$$

where, D = Data; T_t = Transmission time; R_t = Ring latency; N_t = Number of stations

7.5 ISO/OSI STACK

There are two reference models based on the network architectures. One is the International Standards Organization Open Systems Interconnection (Reference Model 1984) (ISO/OSI) model and other is the Transmission Control Protocol/Internet Protocol (TCP/IP) model. The layered architecture of both the models has been compared in Fig. 7.2.

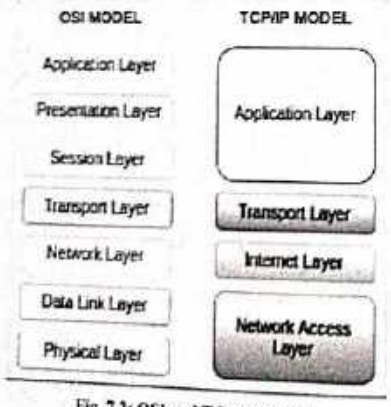


Fig. 7.2: OSI and TCP/IP Model

The ISO/OSI model has seven layers while TCP/IP has only four layers. Session and presentation layers are completely missing. Characteristics of the session layer are provided by

the transport layer, and the application layer bears the accountability of the presentation layer. Host to network is the lowest layer in the TCP/IP model and its duty is to send IP packets to the network. In the ISO/OSI model, the network layer provides connection oriented as well as connectionless services, but the TCP/IP model provides only connectionless services. If M is a message and H is the header that is added at every layer and N layers are present in hierarchy, then the fraction of header that is passed in the total content is calculated as follows:

$$\text{Fraction of data} = \frac{M}{NH + M}$$

7.5.1 ISO/OSI Model

7.5.1.1 Layer 1: Physical Layer

The major responsibilities of physical layer are transmission of raw bit stream and to form the physical interface between two communicating devices. The conversion of analog to digital and digital to analog is performed at this layer.

The following are some issues listed which may create problems before/during transmission:

- Compatibility of mechanical and electrical interfaces
- Working of physical transmission media
- Deciding on the number of bits per second to be sent
- Finding out whether transmission is simplex or duplex
- Establishing and terminating the initial communication when both sender and receiver are finished

$$\text{Transmission time} = \frac{\text{Message size (bits)}}{\text{Bandwidth (bits per second)}}$$

$$\text{Propagation time} = \frac{\text{Distance}}{\text{Velocity}}$$

7.5.1.2 Layer 2: Data Link Layer

Data link layer provides reliable transfer of information between two adjacent nodes. Also, it provides frame-level error control and flow control. It provides communication between machines on the same network. Communication between two devices can be simplex, half duplex, or full duplex. In simplex, the communication is unidirectional. In half duplex, each device can both transmit and receive, but not at the same time. In full duplex, both devices can transmit and receive simultaneously.

This layer is responsible for encoding and decoding, *i.e.*, converting bits to signals at sender site and recovering bits from received signals at receiver side; frame creation, *i.e.*, deciding a minimum unit for sending bits; error detection and/or correction of frames through parity or CRC and flow control using ARQ, sliding WINDOW, etc. The functionality of data link layer is as follows:

- **Encoding:** Signals propagate over a physical medium –
  - modulate electromagnetic waves (varying voltage);
  - encode binary data onto signals (e.g., 0 as low signal or non-return to zero, NRZ, and 1 as high signal or non-return to zero inverted, NRZI); make a transition from current signal to encode a 1 or stay at the current signal to encode a 0;
  - Manchester (transmit XOR of the NRZ-encoded data and the clock only 50% efficient). In Manchester encoding, a clock signal and data signal are mixed together by XORing operation. The clock makes a clock transition in every bit time, so it runs at twice the bit rate. When it is XORed with 0, then it makes low-to-high transition; it acts as a clock and when it is XORed with 1, then it makes high-to-low transition; it acts as a data signal.
- **Framing:** The basic data unit at the date link layer is called a 'frame' which is a collection of bits in sequence boundary. In order to mark boundaries of frame, starting and ending characters are used.
- **Flow Control:** It is a mechanism which informs the sender about the amount of data transmission before receiving an acknowledgement from the receiver. As the receiving device has limited speed for processing the incoming data and limited memory to store data, it informs the sending device by sending few frames and stop. The receiving device has a buffer, a block of memory, for storing extra incoming data before processing.
- **Error Control:** It is a mechanism which informs the sender about the retransmission of the damaged and lost frames during transmission. It is a method of error detection and error correction. Automatic repeat request (ARQ) is a process in which whenever an error is detected, the receiving device sends the request for retransmission to sender.
- **Techniques of Flow and Error Control:**
  - **Stop-and-wait automatic repeat request:** In this protocol, the sender starts the timer and keeps the copy of the sent frame. If the timer expires and there is no acknowledgement (ACK) for the sent frame, the frame is resent, the copy is held and the timer is restarted. For the corrupted and lost ACK frame, sequence numbers can be used. In the data frame, a field is added for the sequence number. The sequence numbers are based on modulo-2 arithmetic. This protocol is also having acknowledgement numbers, which specifies the sequence number of the next frame expected by the receiver. A data frame uses a sequence number and an ACK frame uses an acknowledgement number. The control variable of sender keeps the sequence number for the next frame to be sent (0 or 1). The control variable of receiver keeps the number of the next frame expected. When a frame is sent, the value of the control variable of sender is incremented. When a frame is received, the value of the control variable of receiver is incremented. The stop-and-wait ARQ protocol is very inefficient if the channel is thick (large bandwidth

and long (long round-trip delay). Other drawback of this protocol is that it does not support pipelining, as it does not support multiple frames. Pipelining helps in improving the efficiency of the transmission, if the number of bits in transition is large with respect to the bandwidth-delay product.

For stop-and-wait ARQ,

$$\text{Transmission time } (T_t) = \frac{\text{Message size}}{\text{Bandwidth}}$$

$$\text{Propagation delay } (P_d) = \frac{\text{Distance}}{\text{Velocity}}$$

Link utilization of sender or throughput is given by:

$$\text{Throughput} = \frac{T_t}{T_t + 2 * P_d}$$

- **Go-Back-N automatic repeat request:** This protocol helps in improving the efficiency of transmission by filling the pipe. It supports multiple frames during wait for acknowledgement. The sequence numbers are modulo 2m, where m is the size of the sequence number field in bits. Sliding window defines the range of sequence numbers related to the sender and receiver. When the timer expires, the sender resends all outstanding frames. Stop-and-wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1. It supports one receiver window size. This protocol is very inefficient for a noisy link. In noisy link, frames are resending again and again, which uses bandwidth and slow down the transmission.

  For Go-Back-N ARQ, Sender window size $< 2m$, Maximum sequence number $= 2m - 1$, where m is the number of segment bits. If maximum sequence number is s, then the number of sequence bits = $\log (s + 1)$.

  $$\text{Transmission time } (T_t) = \frac{\text{Message size}}{\text{Bandwidth}}$$

  $$\text{Propagation delay } (P_d) = \frac{\text{Distance}}{\text{Velocity}}$$

  $$\text{Throughput} = \frac{T_t}{T_t + 2 * P_d}$$

  $$\text{Number of frames} = \frac{\text{Total bits}}{\text{Frame size}}$$

- **Selective repeat automatic repeat request:** In this protocol, the damaged frame is resent in the network. It is efficient for noisy links, but it requires complex processing at the receiver end.

Sender window size = Receiver window size $\leq 2m - 1$.

If Q is the size of the window, then the number of sequence bits = $\log_2 Q + 1$

$$\text{Number of frames (Window size)} = \frac{\text{Total bits}}{\text{Frame size}}$$

- **Parity Bits:** Append a single parity bit to a sequence of bits.
  - If using 'odd' parity, the parity bit is calculated as making the total number of 1's in the bit sequence odd;
  - If using 'even' parity, the parity bit makes the total number of 1's in the bit sequence even.
- **Polynomial Codes:** It can detect errors on large chunks of data; has low overhead, is more robust than parity bit, and requires the use of a code polynomial.
- **Cyclic Redundancy Check (CRC):** Example of a polynomial code procedure:
  - Let r be the degree of the code polynomial. Append r zero bits to the end of the transmitted bit string. Call the entire bit string $S(x)$.
  - Divide $S(x)$ by the code polynomial using modulo-2 division. Subtract the remainder from $S(x)$ using modulo-2 subtraction.

  The result is the check summed message.
- **Decoding a CRC Procedure:**
  - Let n be the length of the check summed message in bits.
  - Divide the check summed message by the code polynomial using modulo-2 division. If the remainder is zero, there is no error detected.

Data link layer is divided into two sublayers:

- **Multiple access control sublayer:** It provides controlled access to shared transmission media.
- **Logical link control sublayer:** It is responsible for error and flow control.

When multiple users share a common communication link, multiple access protocols are used to coordinate access to common link (Fig. 7.3).

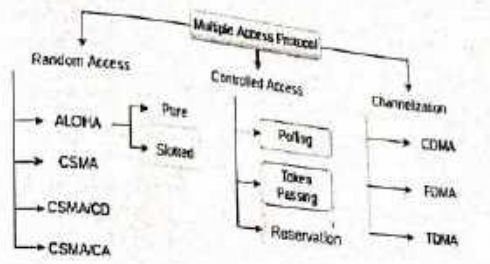


Fig. 7.3: Multiple Access Protocols

1. **Random Access Protocols:** The following are the different random access protocols used for accessing the shared transmission channel:
   - **Pure ALOHA:** It says that whenever a station is having the data, they can send immediately.

     The time at which the collision occurs is called vulnerable time.

     $$T_p = \text{Maximum propagation time} = \frac{\text{Distance between two stations}}{\text{Velocity}}$$

     Suppose T_fr is the average transmission time for a frame, then

     $$T_{fr} = \frac{\text{Framesize}}{\text{Bandwidth}} \quad G$$

     Throughput $(S) = G \times e^{-2G}$

     Vulnerable time $= 2 \times T_{fr}$

     $S_{max} = 18.4\%$
   - **Slotted ALOHA:** It says that if stations are ready with the data, they have to wait for the required time slot and can transmit data exactly at that timeslot.

     Throughput $(S) = G \times e^{-G}$

     Vulnerable time $= T_{fr}$

     $S_{max} = 36.8\%$
   - **CSMA (Carrier Sense Multiple Access):** If a station is ready with the data, it senses the channel, and if channel found idle, data is transmitted, otherwise the station has to wait for random amount of time.
   - **CSMA/CD:** CSMA/CD stands for Carrier Sense Multiple Access/Collision Detection. It is a widely used MAC protocol. Its specifications have been standardized by the IEEE 802.3 standard. When a station wants to transmit a frame

in CSMA, it will sense the channel to check if it is free or not. If the channel is free the station transmits the frame, now it might happen that some other station also transmits the frame, and thus, a collision occurs. Now, the station has to wait(round-trip propagation delay) for an acknowledgement after the transmission in case of CSMA. But with CSMA/CD this time can be reduced.

- **CSMA/CA:** Carrier sense multiple access/collision avoidance (CSMA/CA) is a protocol for carrier transmission in 802.11 networks. It was developed to minimize the potential of a collision occurring when two or more stations send their signals over a data link layer. In this scenario, CSMA requires each station to first check the state of the medium before initiating a transmission. This helps to avert potential collisions by listening to the broadcasting nodes and then informing devices to transmit when the channel is free.

2. **Controlled Access:** In the Controlled access technique, all stations need to consult with one another in order to find out which station has the right to send the data. So, here in controlled access protocols only one station can transmit the data-frames at a time, which leads us to a collision-free transmission through the communication channel.

- **Polling:** The polling method mainly works with those topologies where one device is designated as the primary station and the other device is designated as the secondary station. All the exchange of data must be made through the primary device even though the final destination is the secondary device. Thus to impose order on a network that is of independent users, and in order to establish one station in the network that will act as a controller and periodically polls all other stations is simply referred to as polling. The Primary device mainly controls the link while the secondary device follows the instructions of the primary device. The responsibility is on the primary device in order to determine which device is allowed to use the channel at a given time.
- **Token Passing:** In this access control method, all the stations are logically connected to each other in the form of a ring. The access of stations to the transmission link is governed by a token. A station is allowed to transmit a data packet if and only if it possess the token otherwise not. Each station passes the token to its neighboring station either clockwise or anti-clockwise.
- **Reservation:** In this method, a station needs to make a reservation before sending the data. Time is mainly divided into intervals. Also, in each interval, a reservation frame precedes the data frame that is sent in that interval. Suppose if there are 'N' stations in the system in that case there are exactly 'N' reservation minislots in the reservation frame; where each minislot belongs to a station. Whenever a station needs to send the data frame, then the station makes a reservation in its own minislot. Then the stations that have made reservations can send their data after the reservation frame.

3. **Channelization:** In this, the available bandwidth of the link is shared in time, frequency and code to multiple stations to access channel simultaneously.
   - **CDMA:** One channel carries all transmissions simultaneously. There is neither division of bandwidth nor division of time. For example, if there are many people in a room all speaking at the same time, then also perfect reception of data is possible if only two person speak the same language. Similarly, data from different stations can be transmitted simultaneously in different code languages.
   - **FDMA:** The available bandwidth is divided into equal bands so that each station can be allocated its own band. Guard bands are also added so that no two bands overlap to avoid crosstalk and noise.
   - **TDMA:** In this, the bandwidth is shared between multiple stations. To avoid collision time is divided into slots and stations are allotted these slots to transmit data. However there is a overhead of synchronization as each station needs to know its time slot. This is resolved by adding synchronization bits to each slot. Another issue with TDMA is propagation delay which is resolved by addition of guard bands.

7.5.1.3 Layer 3: Network Layer

The network layer is responsible for host-to-host delivery and path selection between end systems (routing). The fragmentation, reassembly and translation between different network types are also performed at this layer. In other words, communication between nodes is possible in different networks through this layer.

Packet delivery can be accomplished by using either a connection-oriented or a connectionless network service. In a connection-oriented protocol, the connection is established before sending the packets. So, route is established before and all the packets have to follow that route, *e.g.*, Frame relay and ATM uses this service.

In connectionless protocols, the network layer protocol treats each packet independently. The packets in a message may or may not follow the same path to their destination. For example, Internet uses this type of service. Switching can be broadly divided into three categories: circuit switching, packet switching and message switching.

The Internet is a global system of computer networks that are interconnected worldwide and all use the standard Internet protocol suite (TCP/IP) for linking.

- **Internet as a Datagram Network:** The Internet, at the network layer, is a packet-switched network. Switching can be generally divided into three broad categories: circuit switching, packet switching, and message switching. Packet switching uses either the virtual circuit approach or the datagram approach. The Internet chooses the datagram approach to switching in the network layer, and uses the universal addresses defined in the network layer to route packets from the source to the destination. Switching at the network layer in the Internet uses the datagram approach to packet switching.

- **Why Internet Uses Connectionless Network?** In a connection-oriented service, the source has to make a connection with the destination before sending a data packet. Only after establishing connection, a sequence of packets from source to the destination can be sent on the same path that is established before in a sequential order. The connection is terminated only when all the packets of a particular message have been successfully. But the communication at the network layer in the Internet is connectionless. The reason is that Internet is made of so many heterogeneous networks that it is almost impossible to create a connection between every source and destination pair without knowing the nature of the networks in advance.

7.6 ROUTING ALGORITHM

When the router receives a packet, which route this packet should follow to reach to destination is an important concern. This is one of the major responsibilities of network layer. This decision is taken by router on the basis of the routing table maintained by it. The algorithm which decides the suitable route is known as routing algorithm. The desirable properties of any routing algorithm are correctness, fairness, stability, robustness and optimality. These algorithms can be broadly divided into two categories:

- **Adaptive algorithms:** The dynamic routing decision depends on the topology and traffic. Furthermore, adaptive algorithms have been divided into three forms:
  - **Centralized:** The decision is taken on the basis of global information and this is performed by a centralized node.
  - **Isolated:** The routing decision is taken based on local information. Generally, routers do not share information with their neighbours.
  - **Distributed:** A combination of local and global information.
- **Non-adaptive algorithms:** The static routing decision is taken in advance and it is downloaded by the routers, that is, never change once initial route has been selected. The properties of non-adaptive routing algorithms are as follows:
- **Optimality principal:** It states that "if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route". In other words, suppose r_1 is the route from I to J and r_2 is rest of the route. Then, if any route is better than r_2, it could have improved the overall optimal route. Hence, r_1r_2 is optimal.
- **Sink tree:** A set of all optimal routes from any source to a fixed destination form a tree called sink tree. Sink trees may be more than one with the same path length. The concern of sink tree here is to help routers find the best path.

In addition to adaptive and non-adaptive categorisation, routing algorithm can be simply categorised into the following:

- Static routing (shortest path and flooding)
- Flow-based routing
- Dynamic routing (distance vector and link state routing)

- Hierarchical routing
- Routing for mobile hosts
- Broadcast routing
- Multi-cast routing

These are explained as follows:

- **Shortest path routing:** This non-adaptive approach is based on the simplest and most widely used principle. Each node is treated as a router and each arc as communication link. To find a path between a pair of routers, the shortest path is chosen. The shortest path is chosen based on the number of hops or geographical distance in kilometres. Dijkstra's and Bellman-Ford's algorithm are the most famous shortest path algorithms.
- **Flow-based routing:** This is a non-adaptive algorithm which uses topology and traffic condition for deciding the route. If traffic on some route is more than average, then the route should be avoided to achieve optimal path.

  If line capacity and flow is given, delay can be determined easily using the following formula:

  $T = 1/(\mu C - \lambda)$

  where $1/\lambda$ is mean packet size in bits, λ is the mean number of packets arrived per second and C is the line capacity in Kbps.
- **Link state routing:** This is simply a modern replacement of distance vector routing. The steps of the algorithm are as follows:
  - Each router discovers the neighbours for their network addresses.
  - Measure delay or cost to each of these neighbours.
  - Construct a packet including network address and delays of all neighbours.
  - Send it to all routers.
  - Find the shortest path to all routers (Dijkstra's algorithm can be used).
- **Hierarchical routing:** In the situation of telephone networks, all above routing algorithms fail because the size of the routing table is too large here. In this routing, routers are divided (placed) into different regions. A router will have the knowledge of other routers in its own region but unaware about the internal structure of other regions. This will reduce the size of the routing table.
- **Broadcast and multi-casting routing:** Broadcasting means sending packets to all other hosts in the network, whereas multi-casting refers to sending packets to a specific group or a fixed number of hosts.

7.7 NETWORK LAYER PROTOCOLS

In the Internet model, or the TCP/IP suite, there are five main network layer protocols: ARP, RARP, IP, ICMP and IGMP.

The main protocol in this layer is IP. It is responsible for host-to-host delivery of packets from a source to destination. IP needs services of other protocols for better network performance. IP needs ARP to find MAC address of the next hop. As IP is an unreliable protocol, it needs ICMP (Internet Control Message Protocol) to handle unusual situations and errors. IGMP (Internet Group Message Protocol) is used for multi-cast delivery.

7.7.1 Internet Protocol (IPv4)

Internet Protocol is a layer-3 protocol of network layer of OSI model. It takes data segments from layer-4 transport layer and divides it into packets. Thus, it encapsulates data units received from the above layer and adds its own header information (Fig. 7.4).

Maximum size of IP header = 60 bytes

Minimum size of IP header = 20 bytes

If the size of header is 32 bytes in IP, calculate the number of option bytes.

Option bytes = 32 – 20 = 12 bytes

IP header details are as follows:

- **Version:** Version number of Internet Protocol is 4 (*e.g.*, IPv4).
- **IHL:** Internet header length indicates the size of header available in the packet.
- **TOS:** Type of service that is provided by the router to the packets such as minimum delay or cost.
- **Total length:** Length of the entire IP packet (including IP header and IP payload).
- **Identification:** If IP packet size is greater than the maximum transmission unit (MTU), it has to be fragmented during the transmission by the router, then all the fragments of the packet contain same identification number to identify original IP packet they belong to.

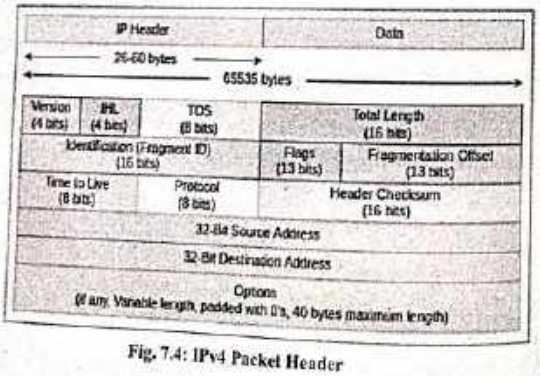


Fig. 7.4: IPv4 Packet Header

- **Flags:** If IP packet is too large, these 'flags' tell if they can be fragmented or not. In 3-bit flag, the MSB is always set to '0'. The next bit is DF (do not fragment). If DF = 0, fragmentation can be done if required and buffered at the router of the receiver until all fragment comes, and if DF = 1, fragmentation should not be done. The third bit is MF (more fragment). If MF = 1, then datagram is not the last fragment. If MF = 0, then datagram is the last fragment.
- **Fragment offset:** This offset tells the exact position of the fragment in the original IP packet.
- **Time to live:** It controls the maximum number of routers visited by the datagram. To avoid looping in the network, every datagram is sent with some time-to-live (TTL) value set. Each router receiving the datagram decreases TTL by 1 and when it becomes 0, the datagram is discarded.
- **Protocol:** It tells the higher-level protocol which uses the service of the IP layer. For example, protocol number for ICMP is 1, IGMP is 2, TCP is 6 and UDP is 17.
- **Header checksum:** This field stores checksum value of the entire header excluding data which is used to check if the packet has been received error-free.
- **Source address:** 32-bit address of the sender of the packet.
- **Destination address:** 32-bit address of the receiver of the packet.
- **Options:** These options may contain values for various options such as strict source routing, security, record route, time stamp, etc.

7.7.1.1 IPv4 Addresses

An IPv4 address is a 32-bit address that can find the device on the Internet uniquely and universally. These addresses are unique, that is, these define only one connection by the device to the Internet. The total number of addresses used by this protocol which is called as address space is 2^n where n is the total number of bits. As IPv4 uses 32-bit addresses, the address space of this protocol is 2^{32}.

In IPv4, addresses are 32-bit binary numbers. However, for ease of use of people, theses binary patters are represented as dotted decimals. Therefore, there are two notations available to denote IPv4 addresses: binary and dotted decimal.

- **Binary notation:** In this notation, IPv4 addresses are represented as 32 bits where each octet is said to be a byte. Therefore, the IPv4 address is usually said to be the 4-byte address. The example for this notation is as follows:

  01111101 10000011 00000110 00000001

- **Dotted-decimal notation:** In this notation, each byte (8 bits) of 32-bit binary address, known as octet is separated with a dot, and then the binary number is converted into its decimal equivalent.

  The example for this notation is as follows:

  11000000 10101000 00001010 00001010 is equivalent to 192.168.10.10

Classful Addressing

The architecture used by IPv4 is classful addressing where the addresses are divided into five classes: A, B, C, D and E. The first few bits reveal the class of address when the address is in binary and first byte reveals the class of address when the address is in dotted decimal notation. This architecture is called classful addressing.

The addresses of class A, B, C are unicast, class D addresses are multi-cast and class E addresses are reserved. The IP address in class A, B and C is divided into NET ID and HOST ID. In class A, one byte defines the NET ID and the other three bytes define the HOST ID. In class B, two bytes define the NET ID, while the other two bytes define the HOST ID. In class C, three bytes define the NET ID and one byte defines the HOST ID.

Mask: The length of the NET ID and HOST ID (in bits) is predetermined in classful addressing but we can also use a mask (also called the default mask) which is a 32-bit number made of contiguous1's. The subnet mask is compared to the IP address from left to right, bit for bit. The 1's in the subnet mask represent the network portion and 0's represent the host portion. The subnet mask is created by placing a binary 1 in each bit position that represents the network portion and placing a binary 0 in each bit position that represents the host portion.

The subnet mask assigned along with IP address signifies which part of the IP address is network and which part is host.

There is a flaw in this type of architecture, that is, each class is divided into fixed number of blocks, and a lot of addresses are wasted as blocks A and B addresses are too large to consume, block C addresses are small in number, class D addresses are reserved for multi-casting and class E addresses are reserved for future, which is another wastage of addresses. Therefore, these address scheme architecture is almost obsolete and leads to an introduction of classless addressing scheme as if there are no address classes.

Classless Addressing

In classless addressing, the addresses are granted in a block and the number of addresses in the block depends upon the addresses needed by the entity. The addresses in the block must be contiguous, the first address must be evenly divisible by the total number of addresses and the number of addresses in a block must be power of 2. Mask in classless addressing can take the value in the range of 0 to 32. Classless Inter-domain Routing provides the flexibility of borrowing bits of host part of the IP address and using them as smaller sub-networks called subnet. This process is known as subnetting. The addresses can be defined in IPv4 as a.b.c.d/n, where a.b.c.d defines one address of the block.

- The first address in the block can be found by setting the rightmost $32 - n$ bits to 0s in the binary notation.
- The last address in the block can be found by setting the $32 - n$ rightmost bits in the binary notation of the address to 1s.
- The number of addresses in the block is the difference between the first and the last address, that is, 2^{32-n}.

Supernetting is a part of classless addressing. In classless addressing, the addresses should be contiguous in a block. The first address should be exactly divisible by the number of addresses in a block. In supernet, bits are borrowed from NET ID.

7.7.2 ICMP

As IP is a connectionless and unreliable protocol, it cannot report errors. So, it takes help of ICMP to communicate updates or error information to other intermediate routers, devices or hosts.

Each ICMP message contains three fields: Type, Code and Checksum (Fig. 7.5). The Type field identifies the ICMP message, the Code field provides further information about the associated Type field and the Checksum field verifies the integrity of the message.

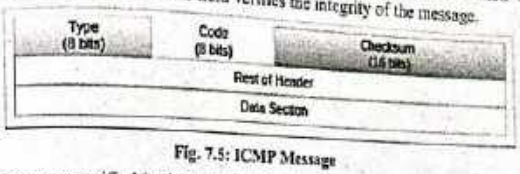


Fig. 7.5: ICMP Message

Extra options are specified in the rest of the header. ICMP places the message to be sent in the data section. Different types defined for ICMP messages are shown in Table 7.1.

Table 7.1: Types of ICMP Messages

Category	Type	Message
Error Reporting Message	3	Destination Unreachable
Query Message	0	Echo Reply
	4	Source Quench
	5	Redirect Message
	11	Time Exceeded
	12	Parameter Problem
	8	Echo Request
	9	Router Solicitation
	10	Router Advertisement
	13	Timestamp Request
	14	Timestamp Reply
	17	Address Mask Request
	18	Address Mask Reply

7.8 TRANSPORT LAYER

Although the reliability of the network layer is undoubtful, the transport layer has many responsibilities to carry out the following:

- Managing connections and timers
- Allowing reliable, connection-oriented byte stream from one end to other end

- Multiplexing
- Addressing
- Performing segmentation
- Packetizing
- Handling error control and variable sized sliding window for flow control
- Allocating bandwidth with congestion control
- Issues
- Headers, error detection, reliable communication
- Communication between processes (running on machines on possibly different networks)

7.9 CONGESTION

Congestion occurs when packets overload the subnet (means the number of packets sent to the network is greater than the capacity of the network), which results in performance degrade. The following are causes for congestion:

- A sudden stream of packet from various sources reaching the same destination.
- Slow links.
- Slow processor.
- Suppose an intermediate router has no free buffer, it will discard the packet. As the sender will not receive any acknowledgement, it will resend the packet and hence congestion will be there.

Congestion is unavoidable, but it is necessary to control it. It can be handled with the following approach:

- Congestion information may be forwarded to the concerned node so that it is possible to limit senders (prevent one sender from overflowing the receiver) or reroute the packets.
- Resource availability may reduce the congestion.
- Prevent additional packets from entering the congestion region.

To control congestion in network traffic, leaky bucket algorithm is used. This algorithm shapes the burst traffic into fixed rate traffic. It does so by averaging the data rate and dropping the packets if bucket is full.

7.9.1 Congestion versus Flow Control

Congestion control ensures the ability to carry the offered traffic by any subnet. The major entities that effect the congestion are behaviour of hosts, routers as well as the factors that reduce the carrying capacity.

Flow control makes it sure that there is not much difference between the sending and receiving packet rate, *i.e.*, a fast sender does not send at a rate faster than the rate at which the receiver receives.

7.10 UDP AND TCP

7.10.1 User Datagram Protocol

UDP, or User Datagram Protocol, is an unreliable and connectionless protocol used by applications that transmit small amount of data at one time and do not require receipt of acknowledgement of data, *e.g.*, audio or video broadcasting.

7.10.2 Transmission Control Protocol

TCP, or Transmission Control Protocol, provides a connection-oriented, reliable stream delivery through the use of sequenced acknowledgement with retransmission of packets when necessary.

7.11 SOCKETS

A socket is an endpoint for communication between client process and server process across a network. A socket address is the combination of an IP address and a port number. This address is used to send data packet to a particular process running on a machine. When two programs are executed, a client process and a server process are created, and these processes communicate with each other by reading from, and writing to, sockets. There are few system calls, such as those given below:

- **Socket()** returns a socket descriptor (like file descriptor), which is an integer value.
- **Bind()** binds an address to a socket descriptor created by socket.
- **Listen()** announces willingness to accept connections.
- **Accept()** blocks the caller until a connection attempt arrives.
- **Connect()** actively attempts to establish a connection.
- **Send()** sends some data over the connection.
- **Receive()** receives some data from the connection.
- **Close()** releases the connection

7.12 SESSION LAYER

The services provided by session layer are as follows:

- Establishes, manages and terminates a communication session with remote systems
- Allows two machines to enter into a dialog (communication may be half duplex or full duplex)

- Adds checkpoints or synchronization points to a data stream.
- Groups several user-level connections into a single 'session'.

Checkpoint: If we need a file of 1000 pages, it is suggested to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. Meanwhile, if a crash occurs during the transmission of page 324, the only pages that need to be resent after system recovery are pages 301 to 324. The pages from 1 to 300 need not be resent.

7.13 PRESENTATION LAYER

The following are the major concerns of presentation layer:

- Syntax and semantics of the information exchanged between two systems.
- Support to different encoding schemes used by different machines. It converts the sender-dependent format into a common format and the receiver converts it back to the receiver-dependent format.
- Data encryption – to ensure privacy, sensitive information should be encrypted. Information encrypted to some other form is unreadable to others.
- Data compression – to reduce the size of information to carry. In the case of multimedia, e.g., text, audio and video, compression is a very useful tool.

7.14 APPLICATION LAYER

The major responsibility of application layer is to implement communication between two applications of the same type. There is a common misconception that every user application runs on application layer, but it runs only on those applications which interact with the communication system. For example, FTP, HTTP and SMTP/POP3/IMAP (email) are all application layer protocols, but a designing software or text editor cannot be considered as an application layer protocol. The following are popular application layer protocols.

7.14.1 Internet Control Message Protocol (ICMP)

ICMP provides error reporting and query management mechanism for host, which lacks in IP. ICMP messages are of two types: error-reporting messages and query messages. The header of ICMP is of 8 bytes and data section is of variable size. First byte is ICMP type, second byte is code, the next two bytes specify the checksum field and rest of the header is specific for each message type.

7.14.2 Domain Name System (DNS)

DNS is a supporting program, which is used by other programs used by the users, such as email. DNS is a client-server program used to find the IP address of an email recipient. In DNS client-server application, a sender sends an email to the email address of the receiver. The DNS client sends the request to the DNS server to map the email address to IP address. DNS has three different domains: generic, country and inverse domains. Generic domain specifies the registered

hosts according to their generic behaviour, e.g., com, org, edu, gov, net, etc. The country domain uses two character country abbreviations, for example, 'in' for India. The inverse domain is used to map an address to a name.

7.14.3 Simple Mail Transfer Protocol (SMTP)

SMTP is a message transfer agent (MTA), which is used to transfer mail. A system should have client MTA for sending a mail and server MTA for receiving a mail. SMTP is used two times, between the sender and the sender's mail server and between the two mail servers. The main task of SMTP is to push the message from the client to the server.

7.14.4 Post Office Protocol (POP)

POP3 (version 3) is a message access protocol which is used to extract the message for client to the server. It has two modes: the delete mode and the keep mode. In the delete mode, the mail is deleted from the mailbox after each retrieval, while in the keep mode, the mail remains in the mailbox after retrieval.

7.14.5 File Transfer Protocol (FTP)

It is used for transferring files from one system to another. FTP establishes two connections between hosts, one for data transfer and the other for control information. FTP uses TCP Port 21 for the control connection and TCP Port 20 for the data connection. The FTP client has three components: user interface, client control process and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection remains open for entire FTP session, whereas the data connection remains open for each file transfer.

7.14.6 Hypertext Transfer Protocol (HTTP)

HTTP is used to access data on the World Wide Web (WWW). It works as a combination of FTP and SMTP. Unlike FTP, HTTP does not have any control connection and uses only one TCP connection. Unlike SMTP, it does not store and then forward the messages. It immediately sends the messages. HTTP uses a TCP Port 80.

7.15 DEVICES

The devices used for internet working at different layers are specified as follows:

- **Repeaters:** It is an electronic device which can receive the weak signal and retransmit it with higher speed. For example, if the LAN is connected for a long distance, then to cover a distance the signal is sent to a repeater which is attached with two LANs and retransmits the signal to a higher level. Thus, repeater connects two segments of network cable. It works at the physical layer of the OSI model.
- **Bridge:** There is a limited number of stations that can be connected with a single LAN. So, a bridge is used to connect multiple LANs of the same type. It operates on a physical layer and data link layer. A bridge checks the physical address contained in

the frame. It also uses table for filtering frames. It partitions the collision so that performance increases.

- **Hub:** It is a network device which is used to connect various computers together. Hub is the central connection for all the computers, which connect through Ethernet. Hub can receive and send the information but cannot perform both tasks at the same time. This makes it slower than a switch. It is less expensive and less complex.
- **Switch:** Switch is a small network device used to connect one or more computers through LAN. It is mostly used in home networks. Switches and hubs are used in the same network. Hubs increase the network by providing more ports, and switches divide the whole network into smaller networks. Switching reduces the amount of unnecessary traffic when every port sends the same information. Switch also reduces the possibility of collision in network.
- **Router:** A router is a networking device which takes packets from one network and after analysis sends that packet to another network. When a data packet comes to the router, the router reads the destination address of the packet and sends it to the respective router which contains that destination address (listed in its routing table). A router is more intelligent than a hub because a hub only sends the information between the devices but the router analyzes the packet and then forwards it to the other network. It controls the traffic on the network.
- **Gateway:** It is a networking system capable of interconnecting one or more networks that has different base protocol. Gateway serves as an entry and exit point. Gateway, sometime called as protocol converter, is used in different layers. For example, a gateway can be used to convert a TCP/IP packet to a NetWare IPX packet.

7.16 NETWORK SECURITY

Network security is an activity designed to protect the network and data in terms of usability, reliability, integrity and safety. It targets a variety of threats and prevents them to enter into the network. It is accomplished through hardware and software, *e.g.*, firewall, anti-virus and anti-spyware, cryptography, intrusion prevention systems (IPS) used to identify fast-spreading threats such as zero-day attacks, and virtual private networks (VPNs) used to provide secure remote access. Network security components are as follows:

- **Confidentiality:** It ensures the concealment of data to unauthorised individuals.
- **Integrity:** It ensures that information is changed in a specified and authorised manner. There is no change in content or source by an unintended user.
- **Availability:** It ensures that systems are available for the authorised users.

The trio form the term CIA (Confidentiality, Integrity and Availability).

7.16.1 Cryptography

Cryptography is the science of providing secure communication over insecure channels. It consists of two operations: encryption and decryption. Encryption is the process in which data is ciphering so that only the intended recipient can know the message. Decryption is the process of deciphering the message.

Basically, encryption and decryption are two functions of a cryptographic algorithm mathematically related to each other. A cryptographic algorithm is widely known, but a key, which is used for the encryption/decryption, is kept secret.

Cryptography is of two types: symmetric cryptography and asymmetric cryptography. Both the approaches have their own pros and cons.

7.16.1.1 Symmetric Cryptography

In symmetric key cryptography, a single shared key is used for both encryption and decryption. Symmetric cryptographic primitives use block ciphers, stream ciphers, cryptographic hash functions, and message authentication codes (MACs). Block cipher uses a deterministic algorithm and operates on a block (fixed length of bits) with unaltered transformation. Stream cipher encrypts each bit individually to generate cipher text. Hash functions or one-way hash functions are used to map an arbitrary-length message string to fixed-size message string. The final value is called hash value.

The security of the symmetric key cryptography lies in the secrecy of the shared symmetric key. If the adversary captures the shared secret key, then it affects both confidentiality and authentication of the message. Examples of symmetric key cryptography are Twofish, Serpent, AES (Rijndael), Blowfish, RC4, RC5, 3DES, IDEA, SEAL, SNOW, etc

7.16.1.2 Asymmetric Cryptography

In asymmetric cryptography, a private key is used for the decryption of a message while a public key is used for the encryption. The private key needs to be kept confidential while the public key can be published freely. Asymmetric cryptography is also known as public key cryptography (PKC). PKC was introduced first by Diffie and Hellman in 1976. Public key algorithms are based on mathematical functions rather than substitution and transposition as in symmetric key cryptography. Examples of asymmetric key cryptography are Diffie-Hellman, RSA, Merkle-Hellman, Rabin, McEliece, El Gamal, Elliptic curves, etc.

7.16.2 Digital Signature

It provides the authenticity of the origin of information to the user and verifies the information is intact. Hence, it provides authentication and data integrity. It also provides non-repudiation, which ensures that the sender cannot deny the origin of information. It is based on the public key cryptography concept.

7.16.3 Firewall

It is a device that filters access to the protected network from the outsider network. It is an integrated collection of security measures, which are designed to prevent unauthorised electronic access to a network system. It has a predefined set of rules, which can protect private network from unauthorised access by filtering incoming or outgoing traffic. These predefined set of rules are called firewall policies.

Functions of Firewalls

- Examining the packet header and filtering
- Verifying the IP address or the port
- Granting and denying access

Packets can be filtered on the basis of the following criteria:

- Source IP address
- Destination IP address
- TCP/UDP source port
- TCP/UDP destination port

7.17 SOLVED QUESTIONS

1. Which layer is responsible for delivery from process to process?
   A. Transport
   B. Network
   C. Physical
   D. Data link

   Solution: (A). Transport layer is responsible for end-to-end process communication.

2. The minimum size of an Ethernet frame is _____.
   A. 18 bytes
   B. 56 bytes
   C. 46 bytes
   D. 64 bytes

   Solution: (D). Ethernet header = 18 Bytes [Dest. Mac(6) + Source Mac (6) + Length (2) + CRC (4)]. Minimum Data Portion = 46 Bytes and Minimum Ethernet Frame Size = 64 Bytes.

3. Using a 7-bit sequence number, what is the maximum size (in bits) of the sender and receiver window using stop-and-wait protocols?
   A. 1 and 7
   B. 1 and 1
   C. 7 and 1
   D. 7 and 7

   Solution: (B). Stop-and-wait protocol uses 1-bit sequence.

4. Vulnerable time in CSMA is _____.
   A. It is double the transmission time as it includes the sensing time
   B. Half of average frame transmission time as collisions are less
   C. 2 × (the average frame transmission time)
   D. None of the above

   Solution: (D). Vulnerable time for CSMA is the propagation time T_p needed for a signal to propagate from one end of the medium to the other.

5. Which class of IP addresses is used for multi-casting?
   A. Class A
   B. Class D
   C. Class B
   D. Class C

   Solution: (B). Class D is used for multi-casting.

6. The options field of IPv4 is used for _____.
   A. Time stamping, strict source routing
   B. Loose source routing, strict source routing
   C. Time stamping only
   D. Time stamping, loose source routing, strict source routing

   Solution: (D). It may contain values for various options, such as strict source routing, security, record route, time stamp, etc.

7. In IPv4 header, the _____ field is used to determine to which datagram a newly arrived fragment belongs to.
   A. Fragment offset
   B. Identification
   C. Datagram-id
   D. Time-to-live

   Solution: (B). Identification field is used to identify original IP packet the fragments belong to.

8. Encryption and decryption is the responsibility of _____ layer.
   A. Session
   B. Network
   C. Application
   D. Data link

   Solution: (C). Application layer is responsible for the encryption and decryption processes.

9. Maximum throughput of an ALOHA network is _____.
   A. 18.4%
   B. 35.8%
   C. 36.8%
   D. 50%

   Solution: (C). When $G = 1$, the throughput is increased to the maximum value of 36.8%.

10. A terminal multiplexer has six 1200 bps terminals and 'N' 300 bps terminals connected to it. The outgoing line is 9600 bps. What is the maximum value of N?
    A. 4
    B. 6
    C. 8
    D. 12

    Solution: (C). Since there are six 1200 bps terminals, $6 * 1200 + n * 300 = 9600 \Rightarrow n = 8$.

11. The total number of wired links required to establish a fully connected mesh network of 9 nodes will be _____.

A. 36
B. 56
C. 72
D. 64

Solution: (A). Total number of wired links required to establish a fully connected mesh network of n nodes can be calculated as $c = n(n-1)/2$. So, for 9 nodes, total links are 3612.

12. In Ethernet CSMA/CD, the special bit sequence transmitted by media access management for collision handling is called _____.
    A. CRC
    B. Jam
    C. Hamming code
    D. Preamble

Solution: (C). Hamming code is a set of error correction code, which is used to detect and correct bit errors. CRC (cyclic redundancy check) is used to detect data transmission errors.

Preamble is used in network communications for synchronizing transmission time between systems.

13. A Gateway operates at _____ layers.
    A. All layers except physical and application layer
    B. All the seven layers
    C. Only on session, transport and network layers
    D. Same layers on which the switch and bridge operates

Solution: (B). The state of that transaction is partially commit.

14. Which of the following statement is not false?

    A. Time stamp protocols avoid deadlock.
    B. Locking technique is used to avoid deadlock.
    C. 2-phase locking does not provide serializability.
    D. 2-phase deals with input and storing phase.

Solution: (A). Hubs, repeaters (or active hubs) operate at physical layer (1); Bridges, Switches operates at data link layer (2); Routers operate at network layer (3); content switches (or web switches or application switches) operates at layers 4 to 7. Gateway operates at all the seven layers-physical, data link, network, transport (4), session (5), presentation (6), application (7).

15. How many number of parity bit is required in hamming code if message size is 8-bit?
    A. 8
    B. 2
    C. 4
    D. 6

Solution: (A). To calculate the number of parity bits, we have the formula: $2^r \geq m + r + 1$, by putting $r = 1, 2, 3, 4$, $r = 4$ will satisfy the given equality.

16. Which of the following assertions is FALSE about the Internet Protocol (IP)?
    A. It is possible for a computer to have multiple IP addresses.
    B. IP packets from the same source to the same destination can take different routes in the network.

    C. IP ensures that a packet is discarded if it is unable to reach its destination within a given number of hops.
    D. The packet source cannot set the route of an outgoing packet; the route is determined only by the routing tables in the routers on the way.

Solution: (D). The packet source can set the route of an outgoing packet; the route is determined only by the routing tables in the routers on the way.

17. Which of the following functionalities must be implemented by a transport protocol over and above the network protocol?
    A. Recovery from packet losses
    B. Detection of duplicate packets
    C. Packet delivery in the correct order
    D. End-to-end connectivity

Solution: (D). End-to-end delivery is the responsibility of a transport layer.

18. Which of the following is NOT true with respect to a transparent bridge and a router?
    A. Both bridge and router selectively forward data packets.
    B. A bridge uses IP addresses while a router uses MAC addresses.
    C. A bridge builds up its routing table by inspecting incoming packets.
    D. A router can connect between a LAN and a WAN.

Solution: (B). Both router and bridge selectively forward data packets and

both can connect between a LAN and WAN. Routing and bridge builds their routing table by inspecting incoming packets but router and bridge both use MAC address. So, option (b) is correct.

19. The transport layer protocols used for real time multimedia, file transfer, DNS and email, respectively, are
    A. TCP, UDP, UDP and TCP
    B. UDP, TCP, TCP and UDP
    C. UDP, TCP, UDP and TCP
    D. TCP, UDP, TCP and UDP

Solution: (C). Real-time multimedia: UDP (session less protocol, used where fast data transfer is required).

20. Which of the following transport layer protocols is used to support electronic mail?
    A. SMTP
    B. IP
    C. TCP
    D. UDP

Solution: (C). Electronic mail does not require TCP connection between sender and receiver of email.

21. The protocol data unit (PDU) for the application layer in the Internet stack is
    A. Segment
    B. Message
    C. Datagram
    D. Frame

Solution: (B). The protocol data unit (PDU) for Data link layer = Frame; Network layer = Datagram; Transport layer = Segment; Application layer = Message.

22. In the IPv4 addressing format, the number of networks allowed under class C addresses is

A. 214
B. 27
C. 221
D. 224

Solution: (C). Starting with 3-bits (110) reserved to recognize the class. So, the number of networks are 221.

23. Which one of the following is not a client-server application?
A. Internet chat
B. Email
C. Web browsing
D. Ping

Solution: (D). Ping is a command not an application. It is to check connectivity and there is no need to communicate with server.

24. Which of the following system calls results in the sending of SYN packets?
A. Socket
B. Listen
C. Bind
D. Connect

Solution: (D). Connect() is called by the client and connection is established using three-way handshake.

25. Which one of the following uses UDP as the transport protocol?
A. HTTP
B. SMTP
C. DNS
D. Telnet

Solution: (C). DNS uses services of UDP protocol.

Answers:

1. A	2. D	3. B	4. D	5. B
6. D	7. B	8. C	9. C	10. C
11. A	12. C	13. B	14. A	15. A
16. D	17. D	18. B	19. C	20. C
21. B	22. C	23. D	24. D	25. C

7.18 PRACTICE QUESTION SETS

1. Which layer is associated with log in/log out from the network?
A. Transport
B. Data link
C. Presentation
D. Session

2. Which layer is associated with IP addresses?
A. Session
B. Transport
C. Network
D. Data link

3. The size of MAC address and IPv4 address is
A. 8 bits and 24 bits
B. 48 bits and 32 bits
C. 32 bits and 48 bits
D. 24 bits and 32 bits

4. The bit size of cyclic redundancy code in an Ethernet is
A. CRC is not used in Ethernet
B. 24
C. 32
D. 8

5. MTU stands for
A. Minimum Transfer Unit
B. Minimum Telephony Unit
C. Maximum Transfer Unit
D. Memory Transfer Unit

6. Using 7-bit sequence number, what is the maximum size (in bits) of the sender and receiver window using Go-Back-N ARQ?
A. 127 and 1
B. 127 and 127
C. 1 and 127
D. 1 and 1

7. Which protocol is used to provide error reporting services to IP address?
A. ICMP
B. ARP
C. OSPF
D. BGP

8. Which protocol is used to find the physical address for a logical address?
A. ARP
B. IGMP
C. RARP
D. ICMP

9. Which protocol cannot be used to find the logical address if the server is residing out of your network?
A. ARP
B. BOOTP
C. RARP
D. IGMP

10. Which topology requires a central controller or hub?
A. Ring
B. Mesh
C. Bus
D. Star

11. The number of nibbles are reserved in the IP header for header length is ____.
A. 5
B. 2
C. 3
D. 4

12. How many hops can be travelled by the IP packet having TTL value 5?
A. It cannot travel any hop.
B. It can travel only single hop.
C. It can travel five hops.
D. It can travel four hops.

13. What does the acronym ISDN for?
A. Indian Standard Data Network
B. Integrated Services Digital Networks
C. Intelligent Secure Digital Network
D. Intelligent Services Digital Network

14. CRC can detect all burst error of up to M errors, if generator polynomial G(x) is of degree
A. Does not matter
B. M/2
C. M-1
D. M+1

15. Which layers of the OSI reference model are host-to-host layers?
A. Transport, session, presentation, application
B. Session, presentation, application
C. Datalink, transport, presentation, application
D. Physical, datalink, network, transport

16. Infrared signals can be used in a closed area using ____ propagation.
    A. Sky
    B. Line of sight
    C. Ground
    D. Small Signal
17. A bridge has access to ____ address in the same network
    A. Physical
    B. Supernet
    C. Logical
    D. Link state
18. The ____ is a unit to measure the signal strength in wireless networks
    A. Frequency
    B. Attenuation
    C. Bandwidth delay product
    D. Decibel
19. Which one of the following media is multi-drop?
    A. UTP cable
    B. Thick coaxial cable
    C. STP cable
    D. Multi-mode fibre optic cable
20. What is the baud rate of the standard 20 Mbps Ethernet?
    A. 10 megabaud
    B. 30 megabaud
    C. 40 megaband
    D. 20 megabaud
21. Binary symmetric channel uses
    A. half-duplex protocol
    B. full-duplex protocol
    C. simplex protocol

    D. simplex protocol with separate data and control bits
22. Which of the following addresses is used to deliver a message to the correct application program running on a host?
    A. Port
    B. Logical address
    C. DNS address
    D. MAC address
23. Which of the following protocol is used for performing RPCs between applications in a language and system in an independent way?
    A. DHCP
    B. SNMP
    C. SOAP
    D. SMTP
24. Which layer of OSI reference model is responsible for decomposition of messages and generation of sequence numbers to ensure correct re-composition from end-to-end communication in a network?
    A. Physical
    B. Transport
    C. Session
    D. Data link
25. Using the 8-bit sequence numbers, what is the maximum size of the sender and receiver window in selective repeat ARQ?
    A. 128 and 128
    B. 1 and 127
    C. 128 and 127
    D. 8 and 8

26. Vulnerable Slotted ALOHA is
    A. equal to average frame transmission time
    B. half of average frame transmission time
    C. two times the average frame transmission time
    D. none
27. Consider an IP in CIDR notation as 220.19.18.87/ 24. The first and the last address of this network will be?
    A. 220.19.18.24 and 220.19.18.87
    B. 220.19.18.0 and 220.19.18.87
    C. 220.19.18.0 and 220.19.18.123
    D. 220.19.18.0 and 220.19.18.255
28. A receiver receives the IP packet with the first 8 bits as 10100011. The length of the IP packet is
    A. 163 bytes
    B. 127 bytes
    C. 1 byte
    D. 12 bytes
29. The router performs at ____ layer(s).
    A. Only physical
    B. Physical, datalink and network
    C. Physical, datalink and transport
    D. Datalink and network
30. Which of the following is true?
    A. In TCP/IP-based services, the destination address is to be specified only during the initial stage of setup.
    B. Initial setup is required for UDP-based service.

    C. Packet sequencing is not guaranteed in TCP/IP-based services.
    D. Initial setup is not possible in UDP-based service.
31. A hub in the network is
    A. a passive device
    B. an active device
    C. a server that serves every node
    D. a power supply concentrator
32. A sender uses public key cryptography to send a secret message. Which of the following is true?
    A. Sender encrypts using receiver's public key
    B. Sender encrypts using his own public key
    C. Receiver decrypts using sender's private key
    D. Receiver decrypts using own private key
33. Transport layer protocols deals with ____.
    A. application to application communication
    B. process to process communication
    C. node to node communication
    D. man to man communication
34. Which one of the following is a version of UDP with congestion control?
    A. Datagram congestion control protocol
    B. Stream control transmission protocol
    C. Structured stream transport
    D. User congestion control protocol

35. Socket-style API for windows is called ____.
   A. wsock
   B. winsock
   C. wins
   D. sockwi
36. An endpoint of an inter-process communication flow across a computer network is called ____.
   A. Socket
   B. Pipe
   C. Port
   D. Machine
37. User datagram protocol is called connectionless because ____.
   A. All UDP packets are treated independently by transport layer
   B. It sends data as a stream of related packets
   C. It is received in the same order as sent order
   D. It sends data very quickly
38. Which of the following are transport layer protocols used in networking?
   A. TCP and FTP
   B. UDP and HTTP
   C. TCP and UDP
   D. HTTP and FTP
39. ICMP is primarily used for ____.
   A. Error and diagnostic functions
   B. Addressing
   C. Forwarding
   D. Routing
40. The network layer protocol for internet is ____
   A. Ethernet
   B. Internet protocol
   C. Hypertext transfer protocol
   D. File transfer protocol
41. Which one of the following algorithm is not used for congestion control?
   A. Traffic aware routing
   B. Admission control
   C. Load shedding
   D. Routing information protocol
42. A subset of a network that includes all the routers but contains no loops is called ____.
   A. Spanning tree
   B. Spider structure
   C. Spider tree
   D. Special tree
43. Which of the following is not correct in relation to multi-destination routing?
   A. Is same as broadcast routing
   B. Contains the list of all destinations
   C. Data is not sent by packets
   D. There are multiple receivers
44. Which of the following routing algorithms can be used for network layer design?
   A. Shortest path algorithm
   B. Distance vector routing
   C. Link state routing
   D. All of the mentioned
45. In virtual circuit network each packet contains ____.
   A. Full source and destination address
   B. A short VC number
   C. Only source address
   D. Only destination address
46. A 4 byte IP address consists of ____.
   A. Only network address
   B. Only host address
   C. Network address & host address
   D. Network address & MAC address
47. Which one of the following is not a function of network layer?
   A. Routing
   B. Inter-networking
   C. Congestion control
   D. Error control
48. The network layer is concerned with ____ of data.
   A. Bits
   B. Frames
   C. Packets
   D. Bytes
49. Physical or logical arrangement of network is ____.
   A. Topology
   B. Routing
   C. Networking
   D. Control
50. Packets in datagram switching are referred to as ____
   A. Datagrams
   B. Switches
   C. Segments
   D. Data-packets

Answers:

1. D	2. C	3. B	4. C	5. C
6. A	7. A	8. A	9. C	10. D
11. D	12. C	13. B	14. D	15. A
16. B	17. A	18. D	19. B	20. B
21. A	22. A	23. C	24. B	25. A
26. A	27. D	28. D	29. B	30. D
31. A	32. A	33. B	34. A	35. B
36. A	37. A	38. C	39. A	40. B
41. D	42. A	43. C	44. D	45. B
46. C	47. D	48. C	49. A	50. A

❐ ❐ ❐

CHAPTER 8

DATABASE MANAGEMENT SYSTEM

SYLLABUS

Introduction to Databases, ER Diagram, Relational Algebra, Relation Calculus, SQL, Normalization, Transactions, Indexing, Query Optimization.

8.1 INTRODUCTION

Database is a collection of organized information so that it can easily be searched, retrieved, managed and updated. A database management system (DBMS) is a set of programs designed to manage a database. It enables users to store, retrieve and modify information in a database with utmost efficiency along with security features. DBMS is applicable to various day-to-day fields such as transactions in banking; airline/railway/hotel reservations; maintenance of student information in schools/universities; online retails; marketing and sales etc. It also allows its users to create their own databases. Different types of DBMS are available such as hierarchical, network, relational and object-oriented.

8.1.1 Traditional File Processing Approach

File processing approach is generally more accurate and faster than the manual database system. Each user is responsible for the defining and implementation of the files required for the specific application. The implementation of required files sometimes creates redundancy of data, for example, one user keeps records for the savings account of the customer and another user may create the loan account of the same customer. This causes the duplicity of the records of the same customer. So, this practice is not feasible for real time applications. There is a need of centralized management of data. The data is created once and then accessed by different users. The data should be shared for different transactions. It should be self-describing in nature, which means the database system contains not only data but it describes the description of the database structure.

8.1.2 Database Management System

A database is a collection of related data. Data is a collection of raw facts or figures, processed to form information. Database management system is a collection of programs for the

creation and maintenance of database. It is an efficient and reliable approach to retrieve data for many users. It provides various functions such as:

(a) **Redundancy control:** It provides redundancy by removing duplicity of data by following rules of normalization.

(b) **Data independence:** It provides independence to application programs from details of data representation and storage. It also provides an abstract view of the data to insulate application code from such details.

(c) **Data integrity:** It promotes and enforces some integrity rules for reducing data redundancy and increasing data consistency.

(d) **Concurrency control:** It supports sharing of data. So, it has to provide an approach for managing concurrent access of the database, hence preserving the inconsistent state and integrity of the data.

(e) **Transaction management:** It provides an approach to ensure that either all the updates for a given transaction will execute or that none of them would execute.

(f) **Backup and recovery:** It provides mechanisms for backing up data periodically and recovering from different types of failures, thus, preventing loss of data.

(g) **Non-procedural query language:** It provides with query language for retrieval and manipulation of data.

(h) **Security:** It protects unauthorized access in the database. It ensures the access to authorized users.

8.2 COMPONENTS OF DATABASE SYSTEMS

DBMS consists of several components, namely software, hardware, data, procedures and data access language. These components are responsible for the definition, collection, management and use of data within the environment. Figure 8.1 shows the components of database system.

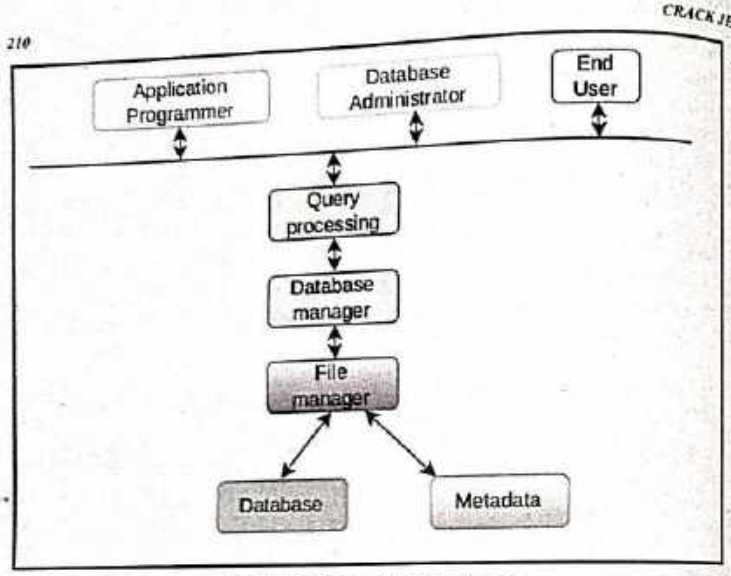


Fig. 8.1: Layers of a Computing System

The description of each component is as follows:

A. Software: It is the collection of programs used by the computers within the database system. It is used to handle, control and manage the database. It includes the following software:

- Operating system software like Microsoft Windows, Linux OS, Mac OS, etc.
- DBMS software such as Oracle 8i, MySQL, Access (Jet, MSDE), SQL Server, etc.
- Network softwares are used for sharing the data of database among multiple users.
- Application programs are developed like C++, VB, .NET, etc. and are used to access database in DBMS. These are used to access and manipulate the data in the database.

B. Hardware: It consists of all system's physical devices such as computers, storage devices, I/O channels, electromechanical devices, etc. It also includes peripherals such as keyboard, mouse, modem, printer, etc.

C. Data: It is the collection of facts. The database contains the data and the metadata.

D. Procedures: These are the instructions and rules to design and use the database system. These includes the following:

- Steps for the installation of DBMS
- Step to use the DBMS or application program
- Steps for the backup of DBMS
- Steps to change the structure of DBMS
- Steps for the generation of reports

E. Data access language: The users can use it to access the data to and from the database. The function of data access language is the entry of new data, manipulation of the existing data and the retrieval of the existing data in the database. The most popular database access language is SQL (Structured Query Language). Users can perform these functions with the help of commands. The role of administrator is to access, to create and to maintain the database.

F. People: Persons involved to access, to create and to maintain the database are called users. These are of various types according to the role performed by them. These are as follows:

- **System Administrator:** The role of system administrator is to supervise the general operations of DBMS.
- **Database Administrator:** The role of database administrator (DBA) is to manage the DBMS.
- **Database Designer:** The role of database designer is to design the structure of the database.
- **Application Programmer:** The role of application programmer is to create the data entry forms, reports and procedures.
- **End-user:** The role of end-user is to use the application programs by entering new data, to manipulating and to access existing data.

8.3 DBMS ARCHITECTURE

It is an approach to outlook the database by users. It is means for the representation of data in an understandable way to the users. DBMS architecture can be used to divide the whole system to related and independent modules. It can be of 1-tier, 2-tier, 3-tier or n-tier.

8.3.1 3-tier Architecture

DBMS can be most widely used as 3-tier architecture. In this architecture, the database is divided into three tiers depending upon the kind of users (Fig. 8.2).

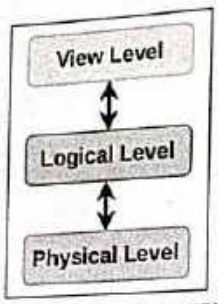


Fig. 8.2: 3-tier Architecture of DBMS

(a) **Internal schema or physical level:** It is also called Database Tier. Database exists in this tier. It also includes query processing languages, all relations and their corresponding constraints. It describes the physical storage structure of the database.

(b) **Conceptual schema or logical level:** It is also called Application Tier. Server and program exists in this tier. It also describes the structure of the database.

(c) **External schema or view level:** It is also called Presentation Tier. End-user exists in this tier. An end-user is capable of using multiple views of database. It also includes all views generated by applications.

8.3.2 Data Independence

Data independence is defined as the change at one level does not affect the higher level. It is of two types:

- **Logical data independence:** It is defined as the change in conceptual schema does not affect the external schema. For example, if the format of the table changes, the data lies in the table should not be changed.
- **Physical data independence:** It is defined as the change in internal schema does not affect the conceptual schema. Thus, it does not affect the external schema. For example, if the storage system has been changed, then it does not affect the logical structure of the database.

8.3.3 Data Models

Data model is a collection of tools, which describes data, relationships, constraints and semantics. It gives the logical structure of the database. It describes the relationship of data in the database. Data models are of various types:

(a) **Relational Model:** It is a collection of relations (tables). In relational model, each table is stored as a separate file.

(b) **Entity-Relationship Data Model:** This model is based on the notion of real-world entities and relationship among them.

(c) **Object-based Data Model:** It defines the database as objects, its properties and its operation. In this model, objects with the similar structures comprise a class. The classes are organized into hierarchies. The operations on these classes are performed through methods.

(d) **Semi-structured Data Model:** It is also known as XML model. This model is used to exchange data over the web. It uses hierarchical tree structures. In this model, data can be represented as elements by using tags.

(e) **Network model:** In this model, data is represented as record types. The data in this model has many-to-many relationship.

(f) **Hierarchical model:** In this model, the data is represented as a hierarchical tree structure.

8.3.4 Relational Model

The database in relational model is represented as a collection of relations (tables). A relation is a kind of set. It is also a subset of a Cartesian product of an unordered set of ordered tuples. Relational model was proposed by E.F. Codd, which stores data in a tabular form. It consists of a table where rows represent records and columns represent the attributes. It has various terminologies as follows:

(a) **Tuple:** It represents a single row of a table, which contains a single record for that relation.

(b) **Relation instance:** It represents a finite set of tuples in the relational database system.

(c) **Relation schema:** It represents the relation name, *i.e.*, table name, attributes and their names.

(d) **Relation key:** It represents the unique key for the relation or table. Each row has one or more attributes, which can identify the row in the table uniquely.

(e) **Attribute domain:** It represents the predefined value scope of each attribute.

8.3.4.1 Constraints in Relational Model

Constraints are the restrictions that one wishes to apply on database. The following constraints are applied on relational model.

(a) **Key constraints:** Each relation has at least one minimal subset of attributes, which can identify a tuple uniquely.

- No two tuples have identical value for key attributes.
- Key attribute does not have NULL value.

(b) **Domain constraints:** Attributes have specific domain values in real world. For example, value of age can only be positive.

(c) **Referential integrity constraints:** If a relation refers to a key attribute of a different relation, then that key element must exist.

8.3.4.2 Relational Algebra

It is a procedural query language. It takes instances of relations as input and returns instances of relations as output. Operators are used to perform queries.

Fundamental operations of relational algebra are as follows:

(a) **Select:** It is used to select rows from a relation. It is denoted by σ.

Syntax of select $\sigma_p(r)$, r is relation and p is prepositional logic. p uses connectors and operators $\vee, \wedge, \neg, =, >, <, \geq, \leq$.

For example, $\sigma_{empname} =$ "*Pabitra*" (emp).

(b) **Project:** It is used to project columns in a relation. It is denoted by Π. The duplicate tuples are automatically eliminated.

Syntax of projects $\Pi A(r)$, r is relation and A is the attribute name in a relation.

For example, Π empname, sal = (emp)

(c) **Union:** It returns a relation instance, which contains all tuples occurring in the first relation or in the second relation. It is denoted as $R \cup S$, where R and S are two relations. The duplicate tuples are automatically eliminated.

This operation is valid for the following:

- Both relations must have the same number of attributes.
- Attribute domains must be compatible.

(d) **Intersection:** It returns a relation instance, which contains all tuples occurring in both relations. It is denoted as $R \cap S$, where R and S are two relations.

(e) **Set difference:** It returns a relation instance, which contains all tuples that occur in the first relation but not in the second relation. It is denoted as $R - S$, where R and S are two relations.

(f) **Cartesian product:** It returns a relation instance, which contains all the fields of the first relation followed by all the fields of the second relation. It is denoted as $R \times S$, where R and S are two relations.

(g) **Rename:** It returns a relation but without any name. It is used to rename the output relation. It is denoted as ρ.

(h) **Joins:** It returns combined information from two or more relations.

- **Condition joins:** It accepts a join condition c and a pair of relation instances as arguments, and returns a relation instance. It is denoted as $\sigma\, c\, (R \times S)$.
- **Equijoins:** It is a special case of condition joins where the condition c contains equalities.
- **Natural join:** It is a Cartesian product of two relations. It is denoted by $\triangleright\triangleleft$.

8.3.4.3 Tuple Calculus

It is a non-procedural query language. In this, number of tuple variables is specified. It is represented as $\{t|Condition\}$, where t is a tuple variable and Condition is a conditional expression. Preliminaries of tuple calculus are as follows:

- Constants
- Predicates
- Boolean, and, or, not
- $\exists$, there exists
- $\forall$, for all

8.3.5 ER Model

ER model represents the conceptual view of a database. It describes the relation of data to each other (Figure 8.3). It was developed by Peter Chen in 1976. It views real-world data as systems of entities and relationships. ER model has three basic elements: entity, attribute and relationship. These are discussed in the following sections.

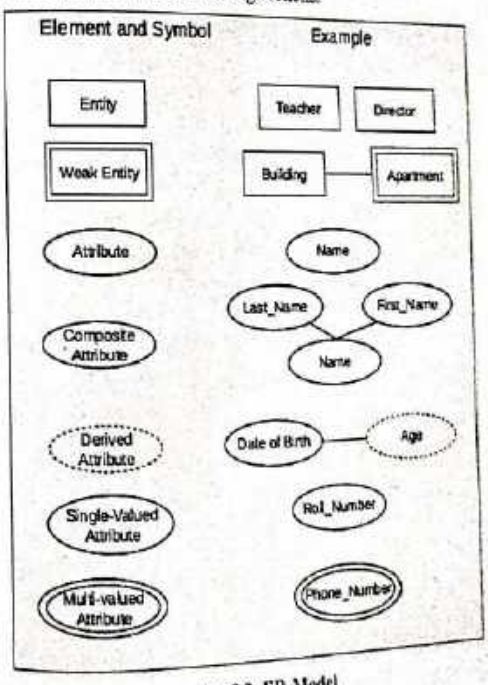


Fig. 8.3: ER-Model

8.3.5.1 Entity

Entities represent the real-world things. These are data objects which maintain different relationships with each other, *e.g.*, Employee, Department, etc. These are represented by means of rectangles.

Entity set is a collection of similar types of entities. For example, all Employees set may contain all employees of all departments.

Weak entity depends on the existence of another entity. A weak entity cannot be identified by its own attributes. It uses a foreign key combined with its attributes to form the primary key. It is represented by means of double rectangles, *e.g.*, details of Employee's spouse, order item, etc.

8.3.5.2 Attributes

Attribute is a property, trait or characteristic of an entity, relationship or another attribute. All attributes have values. A domain or range of values can be assigned to attributes. For example, name, class and age are attributes of the entity student. These are represented by ovals. There are different types of attributes listed as follows:

(a) **Simple attribute:** Simple attributes contains atomic values. Atomic values cannot be divided into sub-parts. Examples are mobile number, roll number, etc.

(b) **Composite attribute:** Composite attributes are composition of many simple attributes. For example, address can be divided into house number, street number, locality and city.

(c) **Derived attribute:** Derived attributes are those whose value is derived from some other attribute in the database. For example, age of person can be calculated from date of birth. Dotted oval is used to represent derived attributes.

(d) **Single-valued attribute:** Single valued attributes contain only one value for that attribute, *e.g.*, age for person, blood group, etc.

(e) **Multi-value attribute:** In this, an attribute may contain more than one value. Multiple values are represented by double ovals, *e.g.*, contact number, email ids, etc.

8.3.5.3 Relationships

It represents the association among entities in a specified way. For example, employee entity has relation works at with department entity. Relationships are represented by diamond-shaped boxes.

Some basic terminologies related to relationship are given below.

As we have discussed, relations are the core components in RDBMS. These relations are defined by two major characteristics – relationship set and the degree of relationship, defined in the following text.

(a) **Relationship Set:** Relationship of similar type is called relationship set. It has attributes. These attributes are called descriptive attributes.

(b) **Degree of Relationship:** It defines the number of participating entities in a relationship. They are of the following types:

- **Unary relationship (Degree 1):** One entity participates.
- **Binary relationship (Degree 2):** Two entities participate.
- **Ternary relationship (Degree 3):** Three entities participate.
- **n-ary relationship (Degree n):** n entities participate.

(c) **Cardinality:** It defines the number of instances of an entity, which can be associated to the number of instances of other entity via relationship set.

- **One to one:** One instance of an entity is associated with at most one instance of another entity with the relationship. It is represented as '1 – 1'.
- **One to many:** One instance of an entity is associated with more than one instance of another entity with the relationship. It is represented as '1 – N'.
- **Many to one:** More than one instance of an entity is associated with another entity with the relationship. It is represented as 'N – 1'.
- **Many to many:** More than one instance of an entity is associated with more than one instance of another entity with the relationship. It is represented as 'N – N'.

(d) **Generalization:** It is a collection of entity sets, having similar characteristics, brought together into one generalized entity. For example, salaried and contract employees are generalized as employee.

(e) **Specialization:** It is the process of identifying subsets of an entity set. It is a reverse process of generalization.

(f) **Aggregation:** It allows a relationship set participate in another relationship set.

8.4 DATABASE DESIGN

The design of a database consists of the following steps:

(a) Identifying entities

(b) Identifying relationships

(c) Identifying attributes

(d) Presenting entities and relationships: Entity relationship diagram (ERD)

(e) Assigning keys

(f) Defining the attribute's data type

(g) Normalization

8.4.1 Integrity Constraints

Integrity constrains are applied to maintain the consistency in a database. This helps in providing the unique answer to a given query on the database.

For example, if the answer to particular query is 'x', then it should be 'x' if such a query is carried out again (without adding/deleting/modifying) on the same table.

(a) **Domain Integrity:** It defines a valid set of values for an attribute, e.g., length or size, data type, etc.

(b) **Entity Integrity Constraint:** It defines that the primary keys cannot be null. There must be a proper value in the primary key field.

(c) **Referential Integrity Constraint:** It is specified between two tables. It is used to maintain the consistency among rows between the two tables.

(d) **Foreign Key Integrity constraint:** There are two types of foreign key integrity constraints:

- **Cascade update related fields:** Whenever the primary key of a row in the primary table is changed, the foreign key values are updated in the matching rows in the related table.
- **Cascade delete related rows:** Whenever a row in the primary table has been deleted, the matching rows are automatically deleted in the related table.

8.4.2 Normal Forms

Normalization is a technique of organizing the database tables, such that they have minimum redundancy. Following are the normal forms that are to be achieved for the process of normalization. The underlying concept is that, if we say a table to be satisfying an 'x' level normal form, then it is understood that it has also satisfied the 'x – 1' level normal form.

(a) **First Normal Form (1NF):** It defines that all the attributes in a relation must have atomic domains.

(b) **Second Normal Form (2NF):** It defines that every non-prime attribute should be fully functionally dependent on prime key attribute. A prime attribute is a part of prime key. The relation must be in the first normal form.

(c) **Third Normal Form (3NF):** It defines that no non-prime attribute is transitively dependent on prime key attribute. The relation must be in the second normal form.

(d) **Boyce-Codd Normal Form (BCNF):** It defines that for any non-trivial FD, $X \rightarrow A$, then X must be a superkey. It is an extension of the third normal form.

(e) **Fourth Normal Form (4NF):** It defines that for every multi-valued dependency $X \rightarrow Y$ that holds over R, one of the following statements is true: (a) Y is the subset of X and (b) X is a superkey. Also, sometimes called, Multi-valued Dependency Normal Form (MDNF).

(f) **Fifth Normal Form (5NF):** It denies that for every join dependency $\triangleright\triangleleft (R_1, \ldots, R_n)$ that holds over R, one of the following statements is true: (a) $R_i = R$ and (b) the join dependency is implied by the set of those FDs over R in which the left side is a key. Also, sometimes called, Project-Join Normal Form (PJNF).

8.4.3 Attribute Closure

Set of all attributes functionally determined by X is called closure of X and is denoted by X^+.

8.4.4 Key

It is a minimum set of attributes used to differentiate all the tuples of the table.

8.4.4.1 Super Key

It is a set of attributes that uniquely identifies each record in a table. It is a superset of candidate key. In other words, let R be the schema and X be the set of attributes over R. If closure of $X(X^+)$ determines all the attributes of R, then X is called superkey.

8.4.4.2 Candidate Key

It is an attribute or set of attribute that can act as a primary key of a table that uniquely identifies each record in a table. Every candidate key is a superkey but not *vice versa*. In other words, candidate key is the minimal superkey. If X is a superkey and none of the proper subset of X is a superkey, then X is called the minimal superkey or candidate key.

8.4.4.3 Primary Key

It is one or more data attributes that uniquely identify an entity. It does not allow null values.

- **Alternate key:** The candidate key, which is not selected as a primary key.
- **Composite key:** It consists of two or more attributes.
- **Foreign key:** It is an entity that is the reference to the primary key of another entity.

8.4.5 Decomposition

When a relation in the relational model is not in appropriate normal form, then the decomposition of a relation is required. In a database, it breaks the table into multiple tables. It is required to eliminate redundancy from the schema. If a relational schema R has redundancy in the data, then decompose R into two R_1 and R_2 schema. There are two properties, which should be maintained when we perform decomposition, it should be a lossless join as well as dependency preserving.

(a) **Lossless Decomposition:** If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.

- It guarantees that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

(b) **Dependency Preserving:** In the dependency preservation, at least one decomposed table must satisfy every dependency.

- If a relation R is decomposed into relation R_1 and R_2, then the dependencies of R either must be a part of R_1 or R_2 or must be derivable from the combination of FDs of R_1 and R_2.
- Consider a relation $R(A, B, C, D)$ with FD set $(A \rightarrow BC)$. The relational R is decomposed into $R_1(ABC)$ and $R_2(AD)$ which is dependency preserving because FD, $A \rightarrow BC$ is a part of relation $R_1(ABC)$.

8.4.5.1 Relation between Two Functional Dependency (FD) Sets

Let F and G be two FD sets:

- Set F and G are equal if and only if closure of F(F') = Closure of G(G').
- Set F and G are equal only if both of the below conditions satisfy:
  - **F covers G:** All FD in G is logically implied by F.
  - **G covers F:** All FD in F is logically implied by G.
- If all FD in G is logically implied by F but all FD in F is not logically implied by G then $G \subset F$.

8.5 TRANSACTIONS AND CONCURRENCY CONTROL

Transactions are series of read and write operations on database. When many users access same database at the same time, some control mechanism is required by databases to stay in consistent state. Following sections will provide description about transactions and their control mechanism.

8.5.1 Transactions

It is a series of reads and writes of database objects. It maintains the integrity of a database while running multiple concurrent operations. Transactions have four properties that a DBMS must ensure to maintain data. These are known as ACID properties:

(a) **Atomicity:** Either all actions are carried out or none (no partial transaction).

(b) **Consistency:** After the execution of transaction, the database must be in a consistent state (no concurrent execution of other transactions).

(c) **Isolation:** All the transactions are carried out and executed as the only transaction in the system (no transaction will affect the existence of any other transaction).

(d) **Durability:** Persistence of data even if the system fails and restarts.

8.5.2 Schedule

A chronological execution sequence of transactions is called schedule. It is a list of actions reading, writing, aborting or committing from a set of transactions. A schedule can have many transactions. Schedule can be further divided into two types: serial schedule and concurrent schedule.

8.5.2.1 Serial Schedule

A schedule is called serial schedule if all transactions are executed in a non-interleaving manner. Let there are two transactions T_1 and T_2 schedule S, then schedule S is called serial schedule if one transaction executes after another.

8.5.2.2 Concurrent Schedule

A schedule is called concurrent schedule if transactions are executed in an interleaving manner or simultaneous execution of two or more transactions. Let there be two transactions T_1 and T_2 in a schedule S, then schedule S is called concurrent schedule if both the transactions execute parallel.

8.6 FILE STRUCTURES

File structure mainly deals with how files are stored on the disk. Some of the important file organizations are described below.

8.6.1 Sequential Files

It is a file organization system where every file record contains an attribute to uniquely identify a particular record. Records are placed in a sequential order with some unique key.

8.6.2 Indexing

Indexing is a data structure mechanism to efficiently retrieve records from the database, for example, book index. It is defined based on its indexing attributes. Indexing is of three types:

(a) **Primary index:** Indexing is based on ordering key field of file.

(b) **Secondary index:** Indexing is based on non-ordering field of file.

(c) **Clustering index:** Indexing is based on ordering non-key field of file.

Ordering field is the field on which the records of file are ordered. Ordered indexing is of two types: dense index and sparse index.

(a) **Dense index:** For every search key value in the database, there is an index record. Index record has two parts: search key value and the pointer. The pointer is pointed to the actual record.

(b) **Sparse index:** In this, no index record is created for every search key.

8.6.3 B-tree

B-tree is a data structure that stores data in such a manner that search, insertions and deletions can be done in *logarithmic* time. B-trees are a general form of binary trees where a node can have more than one child. The B-trees are efficient for those systems that read and write large blocks of data, that is, databases and file systems. B-trees are **self-balancing** trees. All the leaf nodes are at the same level. B-tree with order p has:

(a) Root node that may contain minimum 1 key

(b) Minimum number of child = $(p/2) - 1$

(c) Maximum number of children = p

(d) Maximum keys = $p - 1$

8.7 STRUCTURED QUERY LANGUAGE (SQL)

SQL stands for Structured Query Language, which is a standard computer language for relational database management and data manipulation. It is used to query, insert, update and modify data in the table. Some common RDBMS that use SQL are Oracle, Microsoft SQL Server, Access, Ingres, Sybase, etc. Raymond Boyce and Donald Chamberlin developed it in the early 1970s at IBM, but commercially released by Relational Software Inc. (now, Oracle Corporation) in 1979. Looking into the history it was the software named- VULCAN, that was procured by Ashton Tate and then by FoxPro and later was purchased by Microsoft.

8.7.1 SQL Commands

The SQL commands are used to interact with relational databases.

Some of the important SQL commands with their descriptions are given below:

- **CREATE:** Creates a new table, a view of a table or other objects in the database.
- **ALTER:** Modifies an existing database (table).
- **DROP:** Deletes a table, a view of a table or other objects in the database.
- **SELECT:** Retrieves data from one or more tables (database).
- **INSERT:** Inserts new data record in the database.
- **UPDATE:** Modifies data in the database.
- **DELETE:** Deletes data from the database.
- **GRANT:** Gives a privilege to the user.
- **REVOKE:** Takes back privileges granted from the user.

Some of the important SQL clauses with their descriptions are given below:

- **From:** Equals to cross product.
- **Where:** Selects the tuples which satisfies the condition.
- **Group by:** Groups the table based on specified attribute.
- **Having:** Used to select groups based on condition.

There are four types of language in SQL.

(a) Data Definition Language (DDL): It contains metadata, *i.e.*, data about data. All the integrity constraints and database schemas are defined through DDL. These commands are used to create, modify and delete database objects. CREATE, ALTER, TRUNCATE and DROP are part of DDL. TRUNCATE command is used to delete complete data from an existing table, while DROP command is used to remove a table definition and all data, indexes, triggers and constraints for a table.

- **Syntax Direct Covered**

  CREATE TABLE table_name(column1 datatype, column2 datatype, column3 datatype, ... columnN datatype, PRIMARY KEY(one or more columns));

  For example, create table instructor (INST_ID char(5), name varchar(20), dept_name varchar(20), salary numeric(8,2));

- **Syntax of Alter Command**

  ALTER TABLE table_name ADD column_name datatype;

  ALTER TABLE table_name DROP COLUMN column_name;

  ALTER TABLE table_name MODIFY COLUMN column_name datatype;

  ALTER TABLE table_name ADD CONSTRAINT Constraint UNI (column1, column2...);

  ALTER TABLE table_name ADD CONSTRAINT Constraint CHECK (CONDITION);

  ALTER TABLE table_name ADD CONSTRAINT PrimaryKey PRIMARY KEY (column1, column2...);

  For example, ALTER TABLE CUSTOMERS ADD GRADE char(1);

- **Syntax of truncate command**

  TRUNCATE TABLE table_name;

  For example, TRUNCATE TABLE emp;

- **Syntax of drop command**

  DROP TABLE table_name;

  For example, DROP TABLE emp;

(b) Data Manipulation Language (DML): DML is used to manipulate the data in database. It allows to insert, update and delete data items. SELECT, INSERT, DELETE and UPDATE are part of DML.

- **Syntax of Select Command**

  SELECT * FROM table_name;

  SELECT column1, column2, columnN

  FROM table_name;

  For example, SELECT empno, ename FROM emp;

- Syntax of insert command

  INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)] VALUES

  (value1, value2, value3,...valueN);

  INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

  For example, INSERT into emp (empno, ename, sal, dept) VALUES (100, ABC, 10000, Accounts);

- **Syntax of Delete Command**

  DELETE FROM table_name WHERE [condition];

  For example, DELETE FROM emp WHERE empno = 8;

- **Syntax of Update Command**

  UPDATE table_name SET column1 = value1, column2 = value2...., columnN = valueN

  WHERE [condition];

  For example, UPDATE mp SET sal = 12000 WHERE empid = 8;

(c) **Data Control Language (DCL):** To assign or revoke access rights data control language is used. GRANT and REVOKE are used for DCL.

- **Syntax of Grant Command**

  GRANT privilege_name ON object_name TO user_name | PUBLIC | role_name [with

  GRANT option];

  For example, GRANT SELECT ON emp TO user1;

- **Syntax of Revoke Command**

  REVOKE privilege_name ON object_name FROM User_name | PUBLIC | Role_name;

  For example, REVOKE SELECT ON emp TO user1;

(d) **Transaction Control Language (TCL):** This includes the commands used to manage transactions in the database. For example, COMMIT, ROLLBACK, BEGIN TRANSACTION, SAVEPOINT

8.8 QUERY OPTIMIZATION

Query optimization is the process of selecting an efficient execution plan for evaluating the query. A query is a request for information from a database. A **query plan** (or query execution plan) is an ordered set of steps used to access data in a SQL relational database management system. A single query can be executed through different algorithms or rewritten in different

forms and structures. So, the query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans.

There are two methods of query optimization.

(a) **Cost-based Optimization (Physical):** This is based on the cost of the query. The query can use different paths based on indexes, constraints, sorting methods etc. This method mainly uses the statistics like record size, number of records, number of records per block, number of blocks, table size, whether whole table fits in a block, organization of tables, uniqueness of column values, size of columns, etc.

(b) **Heuristic Optimization (Logical):** This method is also known as **rule-based optimization**. This is based on the equivalence rule on relational expressions. Hence, the number of combination of queries get reduces here. The cost of the query too reduces. This method creates relational tree for the given query based on the equivalence rules. These equivalence rules by providing an alternative way of writing and evaluating the query, gives the better path to evaluate the query. This rule need not be true in all cases. It needs to be examined after applying those rules. The most important set of rules followed in this method is listed below:

  (a) Perform all the selection operation as early as possible in the query. This should be first and foremost set of actions on the tables in the query. By performing the selection operation, we can reduce the number of records involved in the query, rather than using the whole tables throughout the query.

  (b) Perform all the projection as early as possible in the query. This is similar to selection but will reduce the number of columns in the query.

  (c) Next step is to perform most restrictive joins and selection operations. When we say most restrictive joins and selection means, select those set of tables and views which will result in comparatively less number of records. Any query will have better performance when tables with few records are joined. Hence, throughout heuristic method of optimization, the rules are formed to get less number of records at each stage, so that query performance is better. So is the case here too.

  (d) Sometimes we can combine above heuristic steps with cost based optimization technique to get better results.

8.8.1 Purpose of Query Optimization

The major purposes of SQL Query optimization are:

(a) **Reduce Response Time:** The major goal is to enhance performance by reducing the response time. The time difference between users requesting data and getting responses should be minimized for a better user experience.

(b) **Reduced CPU Execution Time:** The CPU execution time of a query must be reduced so that faster results can be obtained.

(e) **Improved Throughput:** The number of resources to be accessed to fetch all necessary data should be minimized. The number of rows to be fetched in a particular query should be in the most efficient manner such that the least number of resources are used.

8.8.2 Various Query Optimization Strategies

Various query optimization strategies are as follows:

(a) Use **Index:** It can be using an index is the first strategy one should use to speed up a query.

(b) **Aggregate Table:** It can be used to pre-populating tables at higher levels. So, less amount of information is required to be parsed.

(c) **Vertical Partitioning:** It can be used to partition the table by columns. This method reduces the amount of information a SQL query required to process.

(d) **Horizontal Partitioning:** It can be used to partition the table by data value, most often time. This method reduces the amount of information a SQL query required to process.

(e) **De-normalization:** The process of de-normalization combines multiple tables into a single table. This speeds up query implementation because fewer table joins are required.

(f) **Server Tuning:** Each server has its parameters and provides tuning server parameters so that it can completely take benefit of the hardware resources that can significantly speed up query implementation.

8.8.3 Important Points on Optimization

The following points should keep in mind for query optimization:

- Avoid using correlated nested queries.
- Avoid inner joins with Equality or OR conditions.
- Check whether records exist before fetching them.
- Use indexing properly, try to create more indexes for fetching composite columns.
- Use Wildcards wisely.
- Try to use WHERE rather than HAVING. Only use HAVING for aggregated values.

8.9 SOLVED QUESTIONS

1. Which of the following operations does not modify the database?
   A. Sorting
   B. Append
   C. Insertion at the beginning
   D. Modify

   Solution: (A). Sorting does not involving addition/insertion or modification of any element.

2. Which of the following operations is based on relational algebra?
   A. Rename and union
   B. Select and union
   C. Only union
   D. Rename, select and union

   Solution: (D). Rename, select and union are some of the operations of relational algebra.

3. ACID properties of a transaction are _____.
   A. Atomicity, constant, integrity, durability
   B. Atomicity, consistency, isolation, durability
   C. Atomicity, compact, in-built, durability
   D. Atomicity, consistency, isolation, database

   Solution: (B). ACID properties are atomicity, consistency, isolation and durability.

4. Which normal form is concerned with removing transitive dependency?
   A. 1NF
   B. 3NF
   C. 2NF
   D. BCNF

   Solution: (B). 3NF removes transitive dependency.

5. The concept Database Lock is used to overcome the problem of _____.
   A. Lost update and inconsistent data
   B. Lost updates, inconsistent data and uncommitted
   C. Inconsistent data only
   D. Uncommitted dependency and inconsistent data

   Solution: (B). Lost update, inconsistent data and uncommitted (WR) problems are overcome by database lock.

6. The concept Database Lock is used to over come the problem of _____.
   A. DML
   B. DDL, DML, query language
   C. Query language
   D. DDL and DML

   Solution: (B). DDL, DML, query language are all database languages.

7. Which of the following is not a function of DBA?
   A. Defining the schema
   B. Network maintenance
   C. Routine maintenance
   D. Data access through authorization

   Solution: (B). The role of database administrator is to manage the DBMS.

8. In a relational database the category type of information is represented in _____.
   A. Tuple
   B. Primary key
   C. Field
   D. Database name

   Solution: (C). Field shows type of information in relational database.

9. Which key is used to identify a tuple uniquely?
   A. Unique key
   B. Tuple key
   C. Primary key
   D. Domain key

   Solution: (C). Primary key.

10. An association of the information is represented by _____.
    A. The records
    B. An attribute

C. A relationship

D. A normal form

Solution: (C). Relationship shows association of information.

11. In which stage of database design, all the necessary fields and their types of a database are listed?

A. E-R diagram

B. Data field definition

C. Data definition

D. User definition

Solution: (C). In data definition stage, all the fields and their types are listed.

12. The maximum length of an attribute of type text is _____.

A. 127

B. 256

C. 255

D. It is a variable

Solution: (D). Text types are Variable length types. For text storage requirements depend on the following factors:

- The Actual length of the column value
- The column's maximum possible

13. When a transaction has completed all its operations and is waiting for commit or rollback action, the state of transaction is _____.

A. Half commit

B. Ready commit

C. Partially commit

D. Commit and rollback

Solution: (C). The state of that transaction is partially commit.

14. Which of the following statement is not false?

A. Time stamp protocols avoid deadlock.

B. Locking technique is used to avoid deadlock.

C. 2-phase locking does not provide serializability.

D. 2-phase deals with input and storing phase.

Solution: (A). Time stamp protocol is a deadlock free protocol.

15. Data integrity control makes use of _____ for maintaining the integrity of database.

A. Integrity Rules

B. Passwords

C. Relational algebra

D. Storing on a backup hard disk

Solution: (A). It promotes and enforces some integrity rules for the reducing data redundancy and increasing data consistency.

16. Data warehouse provides _____.

A. Transaction responsiveness

B. Storage, functionality responsiveness to queries

C. Demand and supply responsiveness.

D. Storage of transactions.

Solution: (B). It provides data storage and responsiveness to queries.

17. If D_1, D_2, ..., D_n are domains in a relational model, then the relation is a table, which is a subset of _____.

A. $D_1 \times D_2, \times \ldots \times D_n$

B. $D_1 \oplus D_2 \oplus \ldots \oplus D_n$

C. $D_1 \cup D_2 \cup \ldots \cup D_n$

D. $D_1 \cap D_2 \cap \ldots \cap D_n$

Solution: (A). $D_1 \times D_2, \times \ldots \times D_n$

18. Which normal form is considered adequate for normal relational database design?

A. 3NF

B. 4NF

C. 2NF

D. 5NF

Solution: (A). Decomposition in BCNF is not always lossless and dependency preserving. So, 3NF is considered to be most adequate form in relational database.

19. Let $R = (A, B, C, D, E, F)$ be a relation scheme with the following dependencies $C \rightarrow F$, $E \rightarrow A$, $EC \rightarrow D$, $A \rightarrow B$. Which of the following is a key for R?

A. EC

B. CD

C. AE

D. AC

Solution: (A). To find key, we try to find closure. $(CD)^+ = \{C, D, F\}$; $(EC)^+ = \{A, B, C, D, E, F\}$; $(AE)^+ = \{A, B, E\}$. Only EC satisfies the closure property.

20.

X	Y	Z
1	4	2
1	5	3
1	6	3
3	2	2

Which of the following functional dependencies are satisfied by the instance?

A. $YZ \rightarrow X$ and $Y \rightarrow Z$

B. $XY \rightarrow Z$ and $Z \rightarrow Y$

C. $YZ \rightarrow X$ and $X \rightarrow Z$

D. $XZ \rightarrow Y$ and $Y \rightarrow X$

Solution: (A). Functional dependency should uniquely identify the value. From the given options, only the following satisfies this condition: $XY \rightarrow Z$, $YZ \rightarrow X$, $Y \rightarrow Z$ and $Y \rightarrow X$.

21. Relation R is decomposed using a set of functional dependencies F, and relation S is decomposed using another set of functional dependencies G. One decomposition is definitely BCNF, the other is definitely 3NF, but it is not known which is which. To make a guaranteed identification, which on one of the following tests should be used on the decompositions? (Assume that the closures of F and G are available.)

A. BCNF definition

B. Dependency preservation

C. Lossless join

D. 3NF definition

Solution: (A). Since to identify BCNF we need BCNF definition. One relation which satisfies will be in BCNF and other will be in 3NF.

Option B is wrong because dependency may be preserved by both 3NF and BCNF.

Option C is wrong Because both 3NF and BCNF decomposition can be lossless.

Option D is wrong because 3NF and BCNF both are in 3NF also.

22. Consider the join of a relation R with a relation S. If R has n tuples and S has m tuples then the maximum and minimum sizes of the join, respectively, are _____.

A. mn and 0

B. $m + n$ and 0

C. $m + n$ and $|m - n|$

D. mn and $m + n$

Solution: (A). Number of tuples in their join gets multiplied with number of tuples in both the relations. So, the maximum and minimum tuples will be mn and 0, respectively.

23. Given the relations Employee (name, salary, deptno) and department (deptno, deptname, address), which of the following queries cannot be expressed using the basic relational algebra operations (σ, π, E, $\bowtie$, $\cap$, $-$):

A. The sum of all employees' salaries

B. Department address of every employee

C. Employees whose name is the same as their department name

D. All employees of a given department

Solution: (A). Aggregate functions such as sum, avg, min and max cannot be expressed in terms of basic relational algebra operations. These require extended relational algebra.

24. Which of the following scenarios may lead to an irrecoverable error in a database system?

A. A transaction reads a data item after it is written by an uncommitted transaction.

B. A transaction reads a data item after it is written by a committed transaction.

C. A transaction writes a data item after it is read by an uncommitted transaction.

D. A transaction reads a data item after it is read by an uncommitted transaction.

Solution: (A). Suppose first transaction A is reading data written by second transaction B, but B has not committed yet. If A acts on that data and eventually commits, but B rollback. A will have "acted on" data that was never committed. So, it will lead to irrecoverable error in database.

25. Consider the following SQL query;

Select distinct $a_1, a_2, \ldots, a_n$

from $r_1, r_2, \ldots r_m$

where P

For an arbitrary predicate P, this query is equivalent to which of the following relational algebra expressions?

A. $\Pi_{a_1, a_2, \ldots, a_n}\, \sigma_p\, (r_1 \times r_2 \times \ldots \times r_m)$

B. $\Pi_{a_1, a_2, \ldots a_n}\, \sigma_p\, (r_1 \bowtie r_2 \bowtie \ldots \bowtie r_m)$

C. $\Pi_{a_1, a_2, \ldots a_n}\, \sigma_p\, (r_1 \cup r_2 \cup \ldots \cup r_m)$

D. $\Pi_{a_1, a_2, \ldots a_n}\, \sigma_p\, (r_1 \cap r_2 \cap \ldots \cap r_m)$

Solution: (A). P is projection that by default selects distinct value of attributes. Cross-product will consider all the tables $r_1, r_2, \ldots, r_m$. Row (σ) selects all the rows satisfying predicate P.

Answers:

1. A	2. D	3. B	4. B	5. B
6. B	7. B	8. C	9. C	10. C
11. C	12. D	13. C	14. A	15. A
16. B	17. A	18. A	19. A	20. A
21. A	22. A	23. A	24. A	25. A

4.10 PRACTICE QUESTION SETS

1. What is the full form of DBMS?

A. Data of Binary Management System

B. Database Management System

C. Database Management Service

D. Data Backup Management System

2. What is a database?

A. Organized collection of information that cannot be accessed, updated and managed

B. Collection of data or information without organizing

C. Organized collection of data or information that can be accessed, updated and managed

D. Organized collection of data that cannot be updated

3. What is DBMS?

A. DBMS is a collection of queries

B. DBMS is a high-level language

C. DBMS is a programming language

D. DBMS stores, modifies and retrieves data

4. Who created the first DBMS?

A. Edgar Frank Codd

B. Charles Bachman

C. Charles Babbage

D. Sharon B. Codd

5. Which type of data can be stored in the database?

A. Image-oriented data

B. Text, files containing data

C. Data in the form of audio or video

D. All of the above

6. In which of the following formats, data is stored in the database management system?

A. Image

B. Text

C. Table

D. Graph

7. Which of the following is not a type of database?

A. Hierarchical

B. Network

C. Distributed

D. Decentralized

8. Which of the following is not an example of DBMS?

A. MySQL

B. Microsoft Access

C. IBM DB2

D. Google

9. Which of the following is a feature of DBMS?

A. Minimum duplication and redundancy of data

B. High level of security

C. Single-user access only

D. Support ACID property

10. Which of the following is a feature of the database?

A. No backup for the data stored

B. User interface provided

C. Lack of authentication

D. Store data in multiple locations

11. Which of the following is not a function of the database?

A. Managing stored data

B. Manipulating data

C. Security for stored data

D. Analyzing code

12. Which of the following is a function of the DBMS?

A. Storing data

B. Providing multi-user's access control

C. Data integrity

D. All of the above

13. Which of the following is a component of the DBMS?

A. Data

B. Data Languages

C. Data Manager

D. All of the above

14. Which of the following is known as a set of entities of the same type that share same properties or attributes?

A. Relation set

B. Tuples

C. Entity set

D. Entity relation model

15. What is information about data called?

A. Hyper data

B. Tera data

C. Metadata

D. Relations

16. What does an RDBMS consist of?

A. Collection of Records

B. Collection of Keys

C. Collection of Tables

D. Collection of Fields

17. The values appearing in given attributes of any tuple in the referencing relation must likewise occur in specified attributes of at least

one tuple in the referenced relation, according to _____ integrity constraint.

A. Referential

B. Primary

C. Referencing

D. Specific

18. _____ is a hardware component that is most important for the operation of a database management system.

A. Microphone

B. High speed, large capacity disk to store data

C. High-resolution video display

D. Printer

19. The DBMS acts as an interface between _____ and _____ of an enterprise-class system.

A. Data and the DBMS

B. Application and SQL

C. Database application and the database

D. The user and the software

20. The ability to query data, as well as insert, delete and alter tuples, is offered by _____.

A. TCL (Transaction Control Language)

B. DCL (Data Control Language)

C. DDL (Data Definition Language)

D. DML (Data Manipulation Language)

21. _____ is a set of one or more attributes taken collectively to uniquely identify a record.

A. Primary key

B. Foreign key

C. Super key

D. Candidate key

22. Which command is used to remove a relation from an SQL?

A. Drop table

B. Delete

C. Purge

D. Remove

23. Which of the following set should be associated with weak entity set for weak entity to be meaningful?

A. Neighbour set

B. Strong entity set

C. Owner set

D. Identifying set

24. Which of the following command is correct to delete the values in the relation EMP?

A. Delete from EMP;

B. Delete from EMP where Id = 'Null';

C. Remove table EMP;

D. Drop table EMP;

25. Procedural language among the following is _____.

A. Domain relational calculus

B. Tuple relational calculus

C. Relational algebra

D. Query language

26. _____ operations do not preserve non-matched tuples.

A. Left outer join

B. Inner join

C. Natural join

D. Right outer join

27. Which forms have a relation that contains information about a single entity?

A. 4NF

B. 2NF

C. 5NF

D. 3NF

28. The top level of the hierarchy consists of _____ each of which can contain _____.

A. Schemas, Catalogs

B. Schemas, Environment

C. Environment, Schemas

D. Catalogs, Schemas

29. _____ indicates the maximum number of entities that can be involved in a relationship.

A. Greater Entity Count

B. Minimum Cardinality

C. Maximum Cardinality

D. ERD

30. The user IDs can be added or removed using which of the following fixed roles?

A. db_sysadmin

B. db_accessadmin

C. db_securityadmin

D. db_setupadmin

31. Why the following statement is erroneous?

SELECT dept_name, ID, avg (salary)

FROM instructor

GROUP BY dept_name;

A. Dept_id should not be used in group by clause

B. Group by clause is not valid in this query

C. Avg(salary) should not be selected

D. None

32. The traditional storage of data organized by the customer, stored in separate folders in filing cabinets is an example of _____ type of "database" management system.

A. Object-oriented database management system

B. Relational database management system

C. Network database management system

D. Hierarchical database management system

33. After groups have been established, SQL applies predicates in the _____ clause, allowing aggregate functions to be used.

A. Where

B. Having

C. Group by

D. With

34. Which of the following is not the utility of DBMS?

(i) Backup

(ii) Loading

(iii) Process organization

(iv) File organization

A. (i), (ii), and (iv) only

B. (i), (ii) and (iii) only

C. (i), (iii) and (iv) only

D. All (i), (ii), (iii), and (iv)

35. What does a foreign key combined with a primary key create?

A. Network model between the tables that connect them

B. Parent-child relationship between the tables that connects them

C. One-to-many relationship between the tables that connects them

D. All of the mentioned

36. Which of the following is correct according to the technology deployed by DBMS?

A. Pointers are used to maintain transactional integrity and consistency

B. Cursors are used to maintain transactional integrity and consistency

C. Locks are used to maintain transactional integrity and consistency

D. Triggers are used to maintain transactional integrity and consistency

37. Which of the following is correct regarding the file produced by a spreadsheet?

A. Can be used as it is by the DBMS

B. Stored on disk in an ASCII text format

C. All of the mentioned

D. None of the mentioned

38. What is the function of the following command? Delete from r where P;

A. Clears entries from relation

B. Deletes relation

C. Deletes particular tuple from relation

D. All of the mentioned

39. _____ resembles Create view.

A. Create table ... as

B. Create view as

C. Create table ... like

D. With data

40. The query specifying the SQL view is said to be updatable if it meets which of the following conditions?

A. Select clause contains relation attribute names but not have expressions, aggregates, or distinct specification

B. From clause has 1 relation

C. Query does not have group by or having clause

D. All of the mentioned

41. Consider a relation $X(p, q, r)$, and the domains are represented by their atomic values. Further, the functional dependency $p \rightarrow q$, $q \rightarrow r$ is observed, the relation is in _____.

A. 1NF, not in 2NF

B. 2NF, not in 3NF

C. 3NF

D. 1NF, not in 3NF

42. Which of the following is/are correct?

A. An SQL query automatically eliminates duplicates

B. An SQL query will not work if there are no indexes on the relations

C. SQL permits attribute names to be repeated in the same relation

D. None of the above

43. A relational database consists of a collection of _____.

A. Tables

B. Fields

C. Records

D. Keys

44. A _____ in a table represents a relationship among a set of values.

A. Column

B. Key

C. Row

D. Entry

45. The term _____ is used to refer to a row.

A. Attribute

B. Tuple

C. Field

D. Instance

46. For each attribute of a relation, there is a set of permitted values, called the _____ of that attribute.

A. Domain

B. Relation

C. Set

D. Schema

47. Database _____ which is the logical design of the database, and the database _____ which is a snapshot of the data in the database at a given instant in time.

A. Instance, Schema

B. Relation, Schema

C. Relation, Domain

D. Schema, Instance

48. Course(course_id, sec_id, semester)
Here, the course_id, sec_id and semester are _____ and course is a _____.

A. Relations, Attribute
B. Attributes, Relation
C. Tuple, Relation
D. Tuple, Attributes

49. A domain is atomic if elements of the domain are considered to be _____ units.
A. Different
B. Indivisible
C. Constant
D. Divisible

50. The tuples of the relations can be of _____ order.
A. Any
B. Same
C. Sorted
D. Constant

Answers:

1. B	2. C	3. D	4. B	5. D
6. C	7. D	8. D	9. C	10. B
11. D	12. D	13. D	14. C	15. C
16. C	17. A	18. B	19. C	20. D
21. C	22. A	23. D	24. A	25. C
26. B	27. A	28. D	29. C	30. B
31. A	32. D	33. D	34. A	35. A
36. C	37. A	38. C	39. A	40. D
41. A	42. A	43. A	44. C	45. B
46. A	47. D	48. B	49. B	50. A

□ □ □

CHAPTER 9

SOFTWARE ENGINEERING

SYLLABUS

Concepts of Networking, Application Areas, Classification, Reference Models, Transmission Environment and Technologies, Routing Algorithms, IP, UDP and TCP Protocols, IPv4 and IPv6, Reliable Data Transferring Methods, Application Protocols, Network Security, Management Systems, Perspectives of Communication Networks.

9.1 INTRODUCTION

Software engineering is the study of design, development and maintenance of software. Real-time problems are so large and complex that they cannot be solved by single developer. Software engineering defines teams and its quality. It is concerned about all the aspects of software development from feasibility of software to its maintenance. The team in software engineering consists of developers, testers, designers, customers, managers, etc. Every team member works to fulfill task assigned to him.

9.2 SOFTWARE

Software is a program, and when it is executed the desired functionality and performance must be satisfied. It is a data structure used to manipulate the information. It is a document that describes the operation and use of a program. Finally, it is a logical entity rather than physical entity.

9.2.1 Characteristics of Software

The following are the characteristics of a software:

(a) Software is developed or engineered, but not manufactured in classical sense. Although the development approach for both the hardware design and software design is same, but after implementation of the hardware design we get the physical component and after implementation of the software design we get the logical

component. In both cases quality is an important factor, that means when the output is operational the associated functionality must be satisfied.

(b) Software does not wear out, but deteriorates. In hardware design, at early development, the failure rate is high due to undiscovered errors. After correcting the defects, the failure rate is decreased and the component starts running at a steady state. However, over a period of time, due to external conditions, such as temperature, air, dust and environmental maladies, the failure rate again increases and the hardware starts to wear out. When it undergoes wear out, we can replace the component with a new one. In software design, at early development, the failure rate is high due to undiscovered errors. After correcting the defects the failure rate is decreased and the software starts running within the constrained level. After a period of time, another change is injected in the existing software due to which the failure rate again increases, and after correction the failure rate decreases. So, during the lifetime of software, when the requirements keep changing, the software undergoes deterioration (Fig. 9.1).

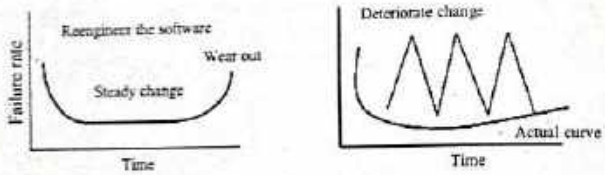


Fig. 9.1: Wear out and Deterioration of Software

(c) The industry is moving towards component-based development, but software is still being custom built. As components are error-free codes, they are reusable. With component-based development for developing some hardware, time and development cost are reduced as the design consists of IC numbers which are already available. So, at the time of implementation we can directly use them. For softwares also, we should use reusable components.

Software development also uses component names such as:

(i) **Off the shelf:** This component is used in the project from the third party, *i.e.*, the component is not available in the library of the developing organization.

(ii) **Fully experienced and partially experienced:** This component is derived based on the previous project. So, it can be directly used in the current project.

(iii) **New component:** This component is developed from the baseline, means from the initial stage or scratch.

9.2.2 Software Engineering

Software engineering is the systematic, disciplined and quantitative approach for the development, operation and maintenance of the software.

Software engineering is described in the following two approaches:

(a) **Layered Approach:** The layered approach consists of the following four layers:

(i) **Quality:** Confirmation to explicitly stated form as mentioned in the document.

(ii) **KPA:** Key process areas such as planning, schedule, performance and quality attribute.

(iii) **Methods:** "How to" process model.

(iv) **Tools:** Automated and semi-automated tool available to develop the software.

(b) **Generic Approach:** The generic approach consists of the following three stages:

(i) **Definition:** This stage is focused on 'What', that is, what information is processed, what functionality and performance is desired, what system behaviour is expected, what constraints are imposed and what validation criteria is used. In this stage, analysis and requirement operations are performed.

(ii) **Validation:** 'Are we building the right product?'. The right condition is injected in the testing phase of the development to cover the errors.

(iii) **Development:** This stage is focused on 'how', how to process the information, how to define the data structure, how to maintain the interface, how to convert procedural description to machine readable code, and how to develop test case etc. This case performs design, coding and testing operations.

9.2.2.1 Support

After delivering, the project support stage is required to maintain the software. The following four types of supports or maintenance are required:

- **Corrective support:** In this support, unidentified errors are going to be managed. The corresponding maintenance is called as corrective maintenance.
- **Adaptive support:** Over a period of time, if the customer wants to change the platform, *i.e.*, CPU, memory, operating system or external interfaces, there is a need of adaptive maintenance.
- **Perfective (enhancement) support:** Over a period of time, if the customer or the end user wants to introduce new functionality into the existing software to maintain the new software, perfective maintenance is required.
- **Preventive support:** When a customer keeps on changing the requirement, the software undergoes deterioration. To maintain such a case, there is a need of preventive maintenance or re-engineering.

9.2.2.2 Software Process

A process gives the framework to develop efficient software. On the basis of the process used to develop the software, the organization undergoes assignment to assess if there is a need of a maturity model.

The Software Engineering Institute (SEI) has introduced a maturity model to assess an organization. This model is called as the capability maturity model (CMM). The CMM consists of the following five (5) maturity levels:

- **Level 0 (Initial):** In this level, there is no standard maintained to develop the software. This level uses the ad hoc procedure, and development becomes chaotic. Success depends on the individual levels.
- **Level 1 (Repeatable):** In this level, based on the knowledge of previous project management, activities are developed to trace the cost, schedule and performance. There is no guarantee to the success scenario of the current project.
- **Level 2 (Defined):** In this level, project management and engineering activities are defined. All the standards are there in the diagrammatical approach, they are not prepared for anticipated situation.
- **Level 3 (Managed):** In this level, project management and engineering activities are quantitatively collected and documented. Quality attributes are also defined. Because of the lack of continuous improvement, minimum quality is assured.
- **Level 4 (Optimized):** In this level, more profit is gained with minimum effort while maintaining the continuous improvement process m.

9.3 PROCESS MODELS

There are two types of process models to develop the software:

9.3.1 Conventional Process Model

In order to develop a software, a team of software engineers follow a development strategy specifying the process, methodology, tools and phases. This is referred to as process model. The different types of software models are as follows:

9.3.1.1 Waterfall Model

Waterfall model, also known as linear sequential model, provides a systematic and sequential approach for the development of a software starting from analysis phase to designing, coding, testing and support.

(a) **Requirement gathering:** On the basis of the business objective, the input domain and output domain are defined. According to the IEEE standard, requirement is defined as a condition or a capability required by the user to interact with the system or "A documented representation of conditional capability is called as requirement".

(b) **Design:** In this stage, the following four operations are performed:

- **Creating the data structure:** It identifies the data objects and attributes and creates the relationship between the data objects.
- **Creating the software architecture:** It creates the blueprint, *i.e.*, the skeleton to represent the inflows and outflows of the software.

- **Identifying the interfaces:** It represents the interconnection between different modules present in the software.
- **Defining procedural details:** It converts the high-level abstraction into low-level abstraction (algorithmic approach).

(c) **Coding:** It translates the procedural details into machine readable form.

(d) **Testing:** In this stage, different test cases are implemented to cover the logical errors and functional errors.

(e) **Supports:** After delivering the software to the customer, support stage is involved to maintain the software and to adapt to the changes.

9.3.1.2 Prototype Model

When customer requirements are not clear, the prototype model is used to finalize the requirement.

In this model, listening to the customer is the entry point to prepare the software requirement specification (SRS). On the basis of the SRS, the designer formulates the quick paradigm of requirements and presents a mockup before the customer. The prototype model is evaluated by the customer, and based on the feedback of the customer its iteration continues until the customer is satisfied. After finalizing the SRS, the same process model is used to build the final product, which is called as open-end prototyping or evaluating prototyping model. After finalising the SRS, the developer can choose another process model to build the final product, which is called as throwaway prototyping or close-end prototyping.

9.3.1.3 Rapid Application Development

The rapid application development (RAD) model is used when the requirements are clear but the time schedule is very short. In this model, the project is divided into modules and develops the model simultaneously. To maintain effective modularity, make sure that the model has **high cohesion and low coupling.**

- **Cohesion:** It is a quantitative measure of functional strength of a module.
- **Coupling:** It is a quantitative measure of to what extent the module is independent from another module.

Framework of the RAD model is as follows:

- **Business modelling:** It identifies the business objective. On the basis of business objective, we can gather data objects.
- **Data modelling:** It creates the data structure, that is, identifies the data objects and attributes and relationship.
- **Process modelling:** It creates the information flow in the project by modelling, adding or deleting the data object.

- **Code generation:** By using a fourth-generation technique, it translates the procedural details into machine-readable format.
- **Test case:** Different test cases are developed and implemented to cover the error. At the end of the test state, all the modules are available with error-free code, which can be integrated to build the final product. RAD model is suitable only when the organization has efficient manpower.

9.3.2 Evolutionary Process Model

In the evolutionary process model, the software is developed in version basis because requirements are not static. When requirement changes, that change is reflected in the version. Under this case, different models are used to develop effective software.

9.3.2.1 Incremental Model

In this model, classic life cycle is used to develop the software.

9.3.2.2 Spiral Model

Steps involved in spiral model are as follows:

- Step 1: Customer communication
- Step 2: Planning
- Step 3: Risk analysis
- Step 4: Engineering
- Step 5: Construction and release
- Step 6: Customer evaluation

(a) In the spiral model, the entry point is the customer communication to finalize the SRS. In the planning stage, we can prepare the estimation model to estimate the cost schedule and performance.

(b) In the risk analysis stage, we can identify the technical and management risks. In the engineering stage, we can select the effective process model to develop the software.

(c) In the construction and release stage, the procedural description is converted into machine readable format and test cases also implemented. Later, the software is delivered to the customer.

(d) In the customer evaluation stage, the software undergoes operation. During operation, the customer identifies the change requirement and posts it to the development team. The change is reflected in the next release of the software.

(e) In the spiral model, customer satisfaction is high, and at the same time the designer is able to design efficient design templates and deliver the code on budget. So, this model is called win-win spiral model.

9.3.2.3 Component-based Development Model

In the component-based model, the components are used in the project development. After designing the project, before constructing the final software check the availability of the components in the library. If the component is available, use it in the current project; if not, construct the new component and place it in the library, then move to the next iteration.

9.4 MEASUREMENT OF METRICS

Metrics is a quantitative measure of an attribute of the project or process.

Measurement is divided into two types:

(a) **Direct measurement:** In the direct measurement, the value is calculated based on direct approach. For example, size of a project can be calculated based on the LOC (line of code).

(b) **Indirect measurement:** Indirect measurement means the value can be calculated based on the other parameters. For example, size of a project can be calculated by using the function point.

Software supports with four kind of P's:

- People
- Product
- Project
- Process

On the basis of the four P's, the software metrics are classified into the following three types:

(a) **Product metrics:** This metrics describes the characteristics of the product such as size, complexity, performance and quality level.

(b) **Project metrics:** It describes the project characteristics and execution. For example, effort cost and schedule; effort in terms of man-months. Effort of a project is 3 man-months means one developer works 3 months to develop the software.

(c) **Process metrics:** It describes the characteristics of the process model to develop and maintain the software (Fig. 9.2). For example, calculates the defect rate and defect removal efficiency. All the estimation models are empirical models, that means, the models are developed based on the past experience without proof. Even though these models lead to success scenario of the software development.

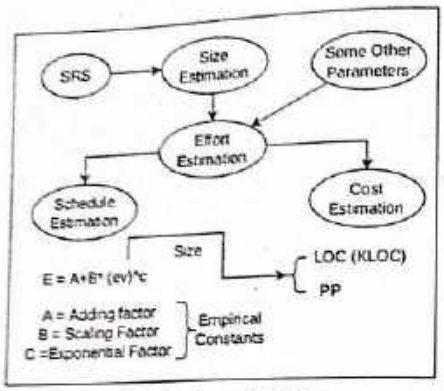


Fig. 9.2: Process Metrics

9.4.1 Size-oriented Metrics

The size of a project can be calculated using the following two ways:

- Line of Code (LOC)
- Function Point Analysis (FPA)

9.4.1.1 Line of Code

If we want to estimate the size of a project, there is a need of taking the past project experience. Assume the LOC into three categories:

- Optimistic $LOC(S_{opt})$
- Most likely $LOC(S_m)$
- Permissible $LOC(S_{pess})$

After identifying three levels of the source code, we can estimate the size of the project by using this following formula:

$$S = \frac{S_{opt} + S_m + S_{pess}}{6}$$

9.4.1.2 Function Point Analysis

Function point analysis (FPA) is an indirect measurement to calculate the size of the project. Therefore, there is a need to consider the other parameters.

The function points are calculated based on the following five attributes:

(a) Number of user input

(b) Number of user output

(c) Number of user enquiries

(d) Number of files

(e) Number of external interface

Every input is associated with the weighing factor. The weighing factor is defined based on the experience of the past project.

If the project is simple, the corresponding values are maintained in a table (Table 9.1). If project is complex, those values are also maintained in table.

Count value = Values × Weighing factor

FP = Count value × VAF

Apart from the functional requirements, some other parameters are also involved in the project.

So, we need to consider the effects of these factors on the project.

$$VAF = 0.65 + 0.01 \times \sum_{i=1}^{4} F_i$$

where VAF is value adjustment factor.

Note: On the basis of the knowledge of the previous project, the weighing table is maintained as a simple/average/complex case. Calculate the count value by providing the multiplication between attribute values with the corresponding complexity levels. Default value of $\sum_{i=1}^{14} F_i = 42$, if value adjustment complexity values are not given, use the default value.

9.4.2 Effort and Schedule (Duration) Estimation

Effort and duration can be estimated by using different empirical model structure.

- **According to the software engineering laboratory (SEL) empirical model:** The effort can be estimated as follows:

  $E = 1.4(KLOC)^{0.93}$ person-months (units)

  The duration can be calculated as $D = 4.6(KLOC)^{0.26}$ months (units)

- **According to Walstonald-Felio method (W-F) (IBM):** The effort can be calculated as:

  $E = 5.2(KLOC)^{0.91}$ person-months

  Duration can be calculated as

  $D = 4.1\ (KLOC)^{0.36}$ months (units)

9.4.2.1 Constructive Cost Model

By using constructive cost model (COCOMO), we can calculate the effort and time, this model is also an empirically derived model. Therefore, the structure of the effort calculation is

$Effort = c_1 EAF\ (size)^{c_2}$ (person/months)

$Duration = c_3\ (Effort)^{c_4}$ (months)

where c_1 and c_3 are the empirically derived scaling factors and c_2 and c_4 are empirically derived exponential factors. Size indicates in terms of KLOC or functional point, EAF (effort adjustment factor) (1 to 14 factors) and rating (1-7) range.

COCOMO model is applicable in three different kinds of applications as follows:

- Organic mode (low complexity)
- Semi-detached mode (average complexity)
- Embedded mode (high complexity)

On the basis of the type of project, the constants are derived as follows:

Table 9.1

	c_1	c_2	c_3	c_4
Simple	2.4	1.05	2.5	0.38
Average	3.0	1.12	2.5	0.35
Complex	3.6	1.20	2.5	0.32

9.4.2.2 Defect Rate

It is a quality metric; defect is undiscovered error during the development of the software. The defect rate is defined as:

Defect rate = (Number of defect × Time)/Total or number of LOC

9.4.2.3 Defect Removal Efficiency

Defect removal efficiency (DRE) is also a quality metric. It is defined as:

$$DRE = \frac{E}{E+D} \times 100$$

where, E = error and D = defect

9.4.2.4 Halstead Size-oriented Metric

Halstead metric is used to count the number of lines preset in the software. To calculate the size of the project in terms of LOC, we are supposed to count the number of lines in the software. But this value does not give the correct estimation because comments and spaces and non-executable lines are also counted.

LOC estimation means only the count of executable statements. To identify the correct executable statement in the software, there is a need to divide the program in the form of lexemes.

Lexemes	Tokens
For	Keyword
=	Equal-op
+	Add-op
*	Mul-op
3	Constant

After dividing the program into token, we can separate the operator and operands. Operators are special words and special symbol operators, etc. Operands are constant, variable, identifier, etc.

Unique Operator	Occurrence of the Operator in the Software	Unique Operand	Occurrence of the Operand in the Software
$\sum = n_1$	$\sum = N_1$	$\sum = n_2$	$\sum = N_2$

where n_1 = count of unique operator used in the software

n_2 = count of unique operands used in the software

N_1 = count of occurrence of the operator in the software

N_2 = count of occurrence of the operator in the software

The following metrics are defined based on the above database.

The vocabulary can be calculated by using $n = n_1 + n_2$

The size of the software can be calculated by $N = N_1 + N_2$

When we are considering machine language, then $N = 2 \times LOC$

The volume of the program can be calculated by using $V = N \log_2 n$

The estimated level of program is defined as $L = \frac{2n_2}{N_1 \times N_2}$

The difficulty is defined as $D = 1/L$

Error is defined as $E = V/L$

Estimated program length is calculated as:

$n_1 \times \log_2 n_1 + n_2 \times \log_2 n_2$

9.5 RISK ANALYSIS

Risk means extreme condition when the product becomes a failure. Risk is associated with two characteristics:

- **Uncertainty:** It means may may not occur, 100% probable risk.
- **Loss:** It means when the risk becomes real there is a loss, otherwise no loss.

9.5.1 Risk Identification

After identifying the extreme condition, there is a need of analyses of the risk in terms of level of uncertainty and degree of loss. The level of uncertainty is maintained in the following four ways:

- Negligible
- Marginal
- Critical
- Catastrophic

We are going to identify critical and catastrophic risk condition and try to manage them in the software execution with the help of risk strategy.

9.5.1.1 Risk Strategy

There are two types of risk strategies:

(a) **Reactive risk strategy:** It means monitoring the project for the probable risk. When the risk is present during execution of the project, it implements the plans to control the risks. It is also called as fire-lighting mode.

(b) **Proactive risk strategy:** It means risk plans and actions are developed before the implementation of the software. So, by using these strategies, we can avoid the risk, if possible, or manage the risk, if not possible. By combining both proactive and reactive strategies, the risk mitigation, monitoring and management (RMMM) plan is prepared.

9.5.1.2 Types of Risk

- **Project risk:** This risk threatens the project plan. Therefore, schedule will be extended and development cost will increase.
- **Technical risk:** These risks threaten the quality and timeliness of developing the software. When this risk becomes real, we cannot develop the project within the quality levels.
- **Business risk:** These risks threaten the viability of software risk. Under this category, four different risks are defined:
  - **Market risk:** Build an excellent product but no users.
  - **Strategic risk:** Build a product that is no longer fit into the business objective.
  - **Management risk:** Lack of high-level support to inject the changes.
  - **Budget risk:** No synchronization between cost and price.
- **Known risk:** Uncover errors after a careful evaluation of the project.

- **Predictable risk:** It is extrapolation from the past project experience.
- **Unpredictable risk:** It acts as a joker in the deck.
- **Generic risk:** It describes the characteristics of the project such as planning, performance, cost, effort, etc.
- **Product-specific risk:** These risks describe the characteristic of the product such as the size, quality, performance, etc.

9.5.1.3 Risk Components

There are four different components present in the software to maintain the probability of probable risk:

- **Performance:** It indicates the level of uncertainty to meet its specification.
- **Cost:** It indicates the level of uncertainty to maintain project budget.
- **Support:** It indicates the level of uncertainty to easily correct, adopt or enhance the existing software.
- **Schedule:** It indicates the level of uncertainty to maintain the schedule and quality of the software.

9.5.2 Risk Projection or Risk Estimation

To estimate the cost of the risk, there is a need to identify the probability of the risk becoming real and the associated impact.

By using the above database, we can estimate the risk cost by using the following formula:

Risk exposure $(RE) = P \times C$

where, P is the probability of risk becoming real and C is the cost.

9.6 SOFTWARE DEVELOPMENT LIFE CYCLE

The software development life cycle (SDLC) describes the generic approach to develop the software. The framework of the SDLC is shown in below:

9.6.1 Requirement

According to the IEEE standard, the requirement is defined as a condition or capability needed by a user to solve a problem or to achieve an objective.

There are five types of requirements classified in the software engineering.

- **Business requirement:** These requirements describe the high-level business requirements. These requirements are documented in vision and scope document.
- **User requirement:** This requirement describes the user need, user goal and user capability towards the system to achieve their objective. These requirements are documented in use-case document.

- **Functional requirements:** This requirement describes the functionality of the software, which is developed into the system to satisfy the user requirement. These requirements are documented in the SRS document.
- **System requirement:** This requirement describes the subsystem requirement on which platform the developed product is running, *i.e.*, CPU, memory capacity and operating system. These are the subset of the functional requirement.
- **Non-functional requirement:** This requirement describes the business rules (such as government acts, government policies and account plans) and quality attributes (availability, usability, maintainability, robustness and flexibility).

9.6.1.1 External Interfaces

Like how many other systems are communicating with the software and constraints (risk condition). All these requirements are documented in the SRS.

Note: SRS document consists of goals of the software, functional requirement and non-functional requirements

In order to develop the right system, the right requirements need to be injected into the software development process, and for developing the right requirement there is a need of requirement engineering.

9.6.1.2 Requirement Engineering

Requirement engineering includes the systematic, disciplined, quantifiable approach to the development, operation and maintenance of the requirement. The following two processes take place in requirement engineering:

- **Requirement management process:** These activities are used to manage the requirement throughout the project development and lifetime of the product.
- **Requirement development process:** The framework used to develop the requirement is shown in Fig. 9.3.

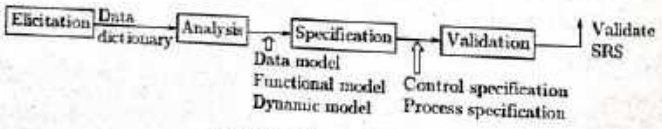


Fig. 9.3: Requirement Process

On the basis of the goal of the software, elicit the requirements from different sources and create the data dictionary.

On the basis of the data dictionary, create the structure of the software. On the basis of the structure of the software, prepare the technical specifications. All the customer capabilities are maintained in the SRS document.

After reviewing the SRS document twice or thrice, the customer and the developer sign on the SRS document. This document (SRS) is then forwarded to the design face.

9.6.1.3 Elicitation

In this state, by using different techniques, elicit the requirement from different sources towards goal of the project. The different elicitation techniques are as follows:

(a) **CORE (Controlled Requirement Expression):** In CORE technique, elicit the requirement by using view point analysis. In this analysis, maintain the sequence of data object, action and consequence.

(b) **IBIS (Issue-based Information System):** In IBIS technique, elicit the requirement based on the question-and-answer analysis. This technique maintains the issue, position and justification.

(c) **FODA (Future-oriented Domain Analysis):** In FODA technique, elicit the requirement by creating different models to specify the structure of the end product. This technique maintains the sequence context model function model dynamic model (behaviour or data model).

(d) **(Quality Function Deployment):** In QFD technique, elicit the requirement towards quality deployment.

(e) **JAD (Joint Application Development):** In JAD technique, elicit the requirement by conducting a quick meeting between customer, end-user, analyst and developers.

(f) **Prototype:** In prototype technique, elicit the requirement by creating the quick design mock-ups and taking continuous feedback from the customer.

Note: The output of the elicitation stage is data dictionary. It consists of data objects related to the software.

9.6.1.4 Analysis

This stage describes the structure of the end product in terms of data object, functionality, information flow and behaviour with respect to external event, by creating three types of model:

1. Context or data model (E-R diagram)
2. Functionality and information flow model (described by DFD)
3. Dynamic or behavioural model (described by STD)

1. **Context or data model (E-R Diagram):** This model describes the structure of the end product in terms of data object and relationship. Data model can be represented by using the entity, relationship diagram. E-R diagram consists of object, attribute and relationship. Object is defined as a representation of composite elements. Composite elements must have multiple properties. The property of the object is called as attribute. The relationship can be maintained between the object by using two concepts.

- **Cardinality:** It indicates in how many ways one object can be communicated with another object. There are three types of cardinality:
  - 1:1
  - 1:n
  - m:n
- **Modelling:** It indicates "is the communication link between the object mandatory or optional".

Note: On the basis of the data model, the data structure is created in the data level design.

2. **Functional and information flow model:** The functional and information flow model describes the functionality of the end product, that means, what is the behaviour of the end product with a proper set of input. The functionality can be represented by using the data flow diagram (DFD). Data flow diagram is also called as Bubble chart or directed graph. Different levels of abstraction of the end product can be represented using different levels of DFD.
   - **Level-Zero DFD:** It describes the high-level abstraction of the functionality by maintaining the single process bubble with multiple input and output-directed edges. In this level of DFD, only the functionality is represented with absence of information flow.
   - **Level-1 DFD:** It describes the functionality and information flow of the software by maintaining different process bubbles. It describes the project in low-level abstraction.
3. **Dynamic model:** It describes the behaviour of the system with respect to external events. This model is represented using the state transition diagram.
   - **Specification:** In this stage, the models are refined into control specification and process specification.
     - **Control specification:** It shows the behaviour of the product.
     - **Process specification:** It refines the mathematical expression, algorithm and constraints.
   - **Validation:** In this stage, various validation criteria are implemented to validate the requirement towards the right condition. The following reviews are done to validate the software:
     - Configuration review
     - Quick review
     - Detailed review

"Validation means the black-box testing is performed at the requirement stage itself". After validating the SRS document, it is forwarded to the design stage. The output of the requirement development process is shown as Fig. 9.4.

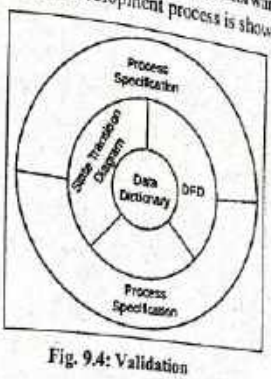


Fig. 9.4: Validation

9.6.2 Design

The design stage represents the project in the low-level abstraction, that means, the customer description is translated into technical description. To do this process, the following four levels of design take place (Fig. 9.5):

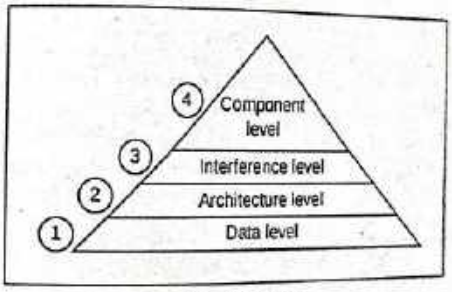


Fig. 9.5: Levels

- **Data level:** Data level design describes the data structure. It takes help from the E-R diagram and data dictionary.
- **Architecture level:** It defines the structure of the end product (skeleton). It takes help from the DFD of the last stage.
- **Interface level:** Interface level design describes the number of interfaces required in the system. It takes help from the state transition diagram.
- **Component level:** It represents the procedural description to implement the functionality in the software. It can take help from process specification and control specification.

9.6.2.1 Concept of Design

The following concepts should be kept in mind while designing a document:

- **Abstraction:** There are different levels of abstraction available to represent the statement. In high level abstraction, the statement is represented in concrete manner, that means, internal structure is not provided. In low level abstraction, all statements are defined in a single format with all details of implementation.
- **Refinement:** It is a top-down strategy to simplify the statement structure. Refinement is also called as elaboration. After refinement of the statement, we can get the direct implementation statement.
- **Modularity:** Software development process uses the modularity concept to reduce the development cost and time. Suppose the complexity of the problem is denoted by $C(P)$ and the effort of the problem is denoted by $E(P)$. The following formats represent the advantage of C:

$C(P_1) = E(P_1)$

$C(P_2) < E(P_2)$

$C(P_1) > E(P_2)$

$C(P_1) > E(P_2)$

$C(P_1 + P_2) > C(P_1) + C(P_2)$

$C(P_1 + P_2) > E(P_1) + E(P_2)$

When the number of modules is less, the cost module for developing the module is high, and at the same time the integration cost is low. When the number of modules increases, the cost per module is low but integration cost is high. Modules should be designed in such a way that each module has specific functional requirements. Functional independence is measured using two terms cohesion and coupling.

9.6.2.2 Cohesion

It is a measure of relative functional strength of a module. Following types of cohesion exist:

- **Logical cohesion:** Logical cohesion exists when a module that performs tasks are related logically.
- **Temporal cohesion:** Temporal cohesion exists when a module contains tasks that are related in such a way that all the tasks must be executed within the same time limit.
- **Procedure cohesion:** The procedural cohesion exists when the processing elements of a module are related and must be executed in a specific order.
- **Communication cohesion:** The communication cohesion exists when all the processing elements concentrate on one area of a data structure.

9.6.2.3 Coupling

Coupling is a measure of relative independence among modules, *i.e.*, it is a measure of interconnection among modules. Following types of coupling exist:

- **Data coupling:** Data coupling exists when a module accesses another module through an argument or through a variable.
- **Stamp coupling:** Stamp coupling exists when a module accesses another module through a data structure.
- **Control coupling:** Control coupling exists when the control is passed between modules using a control variable.
- **External coupling:** External coupling exists when modules are connected to an environment external to the software.
- **Common coupling:** Common coupling exists when the modules share the same global variable.
- **Content coupling:** Content coupling exists when one module makes use of data or control information maintained in another module. Also, content coupling occurs when branches are made into the middle of a module.

Note: Effective modular design must have high cohesion and low coupling.

9.6.2.4 Software Architecture

It shows the hierarchical structure of the software and also describes the logical connections between the modules. In this process, it identifies the data model, functional model and behaviour model.

9.6.2.4.1 Control Hierarchy: It indicates the control sequence in the software project. It represents the interaction between different modules. The control hierarchy defines various parameters of the software design:

- **Depth:** It is a measure of the number of levels present in the software design.
- **Width:** It is a measure of the maximum number of modules present at any one of the level.
- **Fan-in:** It is a measure of the number of modules already controlling the given module.
- **Fan-out:** It is a measure of the number of modules controlled by the given module.
- **Structural partitioning:** It indicates the control sequence of development, maintenance, and integration of the software. It is divided into the following two types:
  - **Horizontal:** Horizontal partitioning allows the level by level development and integration, that means, control flow is assigned to levels.
  - **Vertical:** Vertical portioning means the control sequence is forwarded from root to leaf node, *i.e.*, depth-wise development and integration take place. Because of dependencies, this approach is not suggested to software development.

9.6.2.4.2 Data Structure: It shows the logical connection between the data objects.

9.6.2.4.3 Software Procedure: It describes the customer requirements into programming language statements. Information hiding is used in effective modularity. It shows the independence of the module. After the designing stage, we have the software design document (SDD). The SDD represents the programming description language-oriented customer objective.

9.6.3 Coding

It translates the procedural-oriented description into machine-readable format by using appropriate programming language.

9.6.4 Testing

The objective of testing to measure to what extent the developer logical program becomes a failure. In the testing phase, different test cases are developed and implemented to cover the structural and functional errors. Functional testing includes conducting tests to check the functionality of the product with all proper sets of input, which also covers errors. Structural testing includes conducting tests to examine the internal structure of the developed product to cover the error.

9.6.4.1 Structural Testing

This testing is also called as white-box or glass-box testing. The objective of white-box testing is as follows:

- Guarantee all the statements present in the developed program exercise at least once.
- Exercise logical expression towards true and false side.
- Exercise all the loop boundaries within the operational limit and beyond the operational limit.
- Exercise the data structure present in the program towards validity.

To reach the objectives of white-box testing, two different test operations are performed:

- Basis path testing
- Central structural testing

9.6.4.1.1 Basis Path Testing: This test is used to cover all the paths present in the development code. This testing guarantees that all the statements in the program must be exercised at least once to identify the path in the program, convert the program into graphical notation. On the basis of the control flow between the statements, the program is converted into graph notation called as flow graph or program graph. Flow graph is a directed graph.

Each node in the flow graph represents the program statement. The sequence of the statement is represented by using the following (Fig. 9.6):

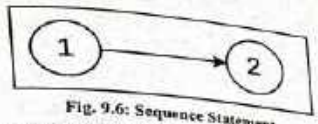


Fig. 9.6: Sequence Statement

The if condition is represented by Fig. 9.7.

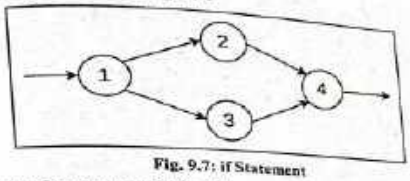


Fig. 9.7: if Statement

The while statement is represented Fig. 9.8.

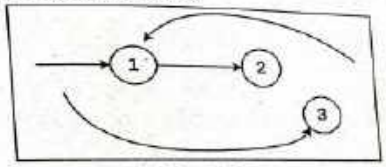


Fig. 9.8: while Statement

Do-while statement is represented Fig. 9.9.

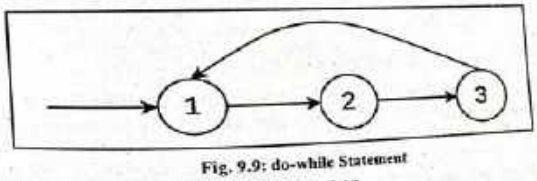


Fig. 9.9: do-while Statement

The switch case statement is represented Fig. 9.10.

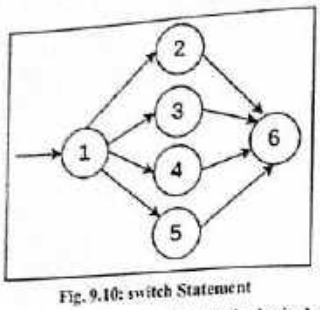


Fig. 9.10: switch Statement

By using the cyclomatic complexity, we can measure the logical complexity of the program.

9.6.4.1.2 Cyclomatic Complexity: It is defined as the quantitative measure of logical complexity of the program. Cyclomatic complexity can be calculated in three ways:

- By using the number of edges and number of nodes we can measure the cyclomatic complexity as:

  $V(G) = E - N + 2P$

  where, E = Number of edges, N = Number of nodes and P = Number of disconnected component

- By using the region, we can estimate the cyclomatic complexity as:

  $V(G)$ = Number of regions present in the graph.

  When we count the cyclomatic complexity of a program by using basis path testing, outside of graph is considered as one region.

- By using predictable nodes:

  $V(G) = P + 1$

  where, P = Number of predicate nodes.

  Predictable nodes are calculated by two or more nodes expanding from it.

9.6.4.1.3 Graph Matrices: The flow graph is converted into graph matrices by maintaining rows and columns. The number of rows and columns depends on the number of nodes in the flow graph. Based on the communication link between the nodes in the flow graph, the entry is reflected in corresponding position in the graph matrices. If any row consists of more than one entry that node is called as predicate node, then take the weightage of predicate node by subtracting one from the total. After subtraction operation, add all the row values, it gives the predicate nodes. Use the formula to calculate cyclomatic complexity:

$V(G) = P + 1$.

9.6.4.1.4 Control Structure Testing: This testing is used to cover the error from the control structure present in the program. There are three types of expressions:

- Arithmetic (+ , - , *, /) = Value
- Relational (>, <, <=, >=, =, =) = True/False (Boolean)
- Boolean (&&, |, ¬, /) = True/False (Boolean data type)

Examples:

- $a + b$
- $(a + b) > (c + d) = T/F$
- $((a + b) > (c + d)\&\&((a \times b) < ((c * d)) = T/F$

After evaluating the control expression, the output may be a Boolean data, that is, true/false. In the control structure testing, examine all the logical expression towards true/false. During that evaluation, it covers the Boolean operator error, Boolean data or parentheses symbol error, relational operator error and arithmetic operator errors.

In this testing, two test cases are developed to cover all the errors described in the above list.

- Branch testing
- Domain testing

9.6.4.2 Black-box Testing

In this testing, various input test cases are developed and the product functionality is examined, whether it satisfies all possible inputs. This testing is also called behavioural testing or grey-box testing. In this strategy, various testing techniques are used to cover the functional errors such as:

- Equivalence partition
- Boundary value analysis
- Comparison testing
- **Equivalence partition:** In this testing, the input domain is divided into equivalent partitions. Each position is called as class. One partition is the exact valid range and another partition is above the upper limit and another is below the lower limit.
- **Boundary value analysis:** Most of the problems are possible at the extreme edges that are boundaries of input domain. Boundary value analysis focuses on identified errors at the boundary rather than finding those existing at the centre of the input domain.
- **Comparison testing:** When the software is developed in the real-time application domain, there is a predefined value estimated based on the product behaviour. After developing the software, compare the behaviour of the new product with predefined values and remove the errors.

9.6.4.3 Life Cycle Testing

Various testing plans are implemented at each stage of the software development life cycle.

9.6.4.4 Validation Testing

This testing is performed at requirement stage by conducting the black-box test plans on a prototype model.

9.6.4.5 Integration Testing

This testing is performed at the design stage. Software development uses the modularity concept and develops the module independently. After developing the module, there is a need of integration to generate the final system.

In that regard, integration testing is required to check the functionality. Integration testing is not focused on internal details of the modules.

9.6.4.5.1 Top-down Integration: This testing focuses on the function of the module integration either depth-wise or breadth-wise. Depth-wise integration is a vertical partitioning and width-wise integration is horizontal partitioning.

9.6.4.5.2 Bottom-up Integration: In this testing, the integration starts from leaf node and forwards to the root node. In this testing, the stubs and drivers are maintained.

9.6.4.6 Regression Testing

When a new module is added, there is a possibility of introducing new functionality to the existing system. Due to the new functionality, there is a possibility of errors occurring in the existing system. To cover those errors, all test cases are re-conducted. This is called regression testing.

9.6.4.7 Smoke Testing

This testing is used in the wrap-shrink component development. Wrap-shrink component development means within the short period of time the software is implemented. To perform this we use modularity and by adding or deleting the existing module, we can develop the final product. In this way, smoke testing is used to cover the error.

9.6.4.8 Unit Testing

This testing is performed at the coding stage. After developing the module, use the white-box test plan to cover the structural errors in the module.

9.6.4.9 Alpha Testing

After developing the product, this test is conducted by the customer at the developer side to cover the error. In this testing, the developer's presence is there.

9.6.4.10 Beta Testing

After deploying the product to the customer side, this test is conducted by the customer or end user at the customer place. In this testing, the developer's presence is not there.

9.6.4.11 System Testing

After developing the product, various system-level testing operations are performed to check the reliability of the product.

(a) **Recovery testing:** This testing focuses on the recovery of data due to crashes or power failure.

(b) **Security testing:** This testing focuses on unauthorised accessing of product.

(c) **Stress testing:** This testing is focused on load balancing. It is also known as volume testing.

(d) **Performance testing:** This testing is a user acceptance testing in terms of quality and reliability

9.7 SOLVED QUESTIONS

1. The most important feature of spiral model is _____.
   A. Risk management
   B. Requirement analysis
   C. Quality management
   D. Configuration management

   Solution: (A). Risk management helps in developing a cost-effective project where risks are analyzed and risk strategies are decided upon.

2. Cyclomatic complexity of a flow graph G with n vertices and e edges is _____.
   A. $V(G) = e + n - 2$
   B. $V(G) = e + n + 2$
   C. $V(G) = e - n + 2$
   D. $V(G) = e - n - 2$

   Solution: (C). Cyclomatic complexity is a software metric developed by Thomas J. McCabe. It is used to indicate the complexity of a program. It directly measures the number of linearly independent paths through a program's source code. It is also known as conditional complexity. It is defined as $V(G) = e - n + 2$.

3. Testing of software with actual data and in actual environment is called _____.
   A. Alpha testing
   B. Regression testing
   C. Beta testing
   D. None of the above

   Solution: (C). Users or an independent test team at the developer's site performs alpha testing. Alpha testing is often employed as a form of internal acceptance testing, before the software goes to beta testing.

   Beta testing is performed after alpha testing for user acceptance testing.

   Regression testing focuses on finding defects after a major code change has occurred.

4. Key process areas of CMM level 4 are also classified by a process which is _____.
   A. CMM level 2
   B. CMM level 5
   C. CMM level 3
   D. All of the above

   Solution: (B). There are five maturity levels in CMM model: (1) Initial,

(2) Managed, (3) Defined, (4) Quantitatively Managed and (5) Optimizing. The organization at level 2 includes level 1. So, level 4 is included in level 5.

5. Validation means _____.
   A. Are we building the right product
   B. Are we building the product right
   C. Certification of fields
   D. None of the above

Solution: (A). Verification means are we building the product right, whereas validation means whether we are building the right product.

6. Function points can be calculated by _____.
   A. UFP × FAC
   B. UFP × CAF
   C. UFP × Cost
   D. UFP × Productivity

Solution: (B). Function Point FP = UFP × CAF. UFP stands for Unadjusted Function Point, while CAF stands for Complexity Adjustment Factor.

7. Superficially, the term "object-oriented" means that we organize software as a _____.
   A. collection of discrete objects that incorporates both discrete structure and behaviour
   B. collection of objects that incorporates both discrete data structure and behaviour
   C. collection of discrete objects that incorporates both data structure and behaviour
   D. collection of continuous objects that in corporates both data structure and behaviour

Solution: (C). This is the definition of object-oriented according to software engineering.

8. Which one of the following is not a key process area in CMM level 5?
   A. Process change management
   B. Technology change management
   C. Software product engineering
   D. Defect prevention

Solution: (C). The key process areas at level 5 covers the following issues: Defect Prevention, Technology Change Management and Process Change Management. Software Product Engineering has been addressed in level 3.

9. Which of the following system components is responsible for ensuring that the system is working to fulfill its objective?
   A. Input
   B. Feedback
   C. Output
   D. Control

Solution: (D). The components of the control system ensure smooth functioning of the system to achieve its objective.

10. Modifying the software to match changes in the ever-changing environment is called _____.
    A. Adaptive maintenance
    B. Corrective maintenance
    C. Perfective maintenance
    D. Preventive maintenance

Solution: (A). Corrective maintenance refers to modifications initiated by defects in the software. Perfective maintenance refers to improving processing efficiency or performance, or restructuring the software to improve changeability. Preventive maintenance refers to prevention of breakdowns.

11. Software measurement is useful to _____.
    (a) Indicate quality of the product
    (b) Track progress
    (c) Form a baseline for estimation and prediction
    (d) All of the above

Solution: (C). Software measurement is useful to indicate quality of the product, track progress, assess productivity and form a baseline for estimation and prediction.

12. Which of the following metric does not depend on the programming language used?
    A. Line of code
    B. Member of token
    C. Function count
    D. All of the above

Solution: (C). Line of code and tokens depends upon the programming language used for developing programs.

13. If in a software project the number of user input, user output, enquiries, files and external interfaces are (15, 50, 24, 12, 8), respectively, with complexity average weighing factor. Find productivity if effort = 70 person-month.
    A. 408.74
    B. 308.74
    C. 208.74
    D. 408.47

Solution: (A). Count value $(15 \times 4) + (50 \times 5) + (24 \times 4) + (12 \times 10) + (8 \times 7) = 382$; Productivity = Count value $\times (0.65 + 42 \times 0.01) = 382 \times (0.65 + 0.42) = 382 \times (1.07) = 408.74$.

14. A process view in software engineering would consider which of the following:
    A. Product performance
    B. Staffing
    C. Functionality
    D. Reliability

Solution: (B). Staffing is the apt choice among the given.

15. A design is said to be a good design if the components are _____.
    A. Strongly coupled
    B. Strongly coupled and weakly cohesive
    C. Strongly coupled and strongly cohesive
    D. Strongly cohesive and weakly coupled

Solution: (D). The aim is to minimize the interaction between modules and maximize interaction within a module.

16. If a control switch is passed as an argument, this is an example of _____ coupling.
    A. Content
    B. Control
    C. Data
    D. Common

Solution: (B). Two modules are control coupled if one passes an element of control to another.

17. The aim of software engineering is to produce software that is _____.
    A. Fault-free
    B. Delivered on time
    C. Delivered within budget
    D. All of these

Solution: (D). The aim of software engineering is to produce software that is fault-free, delivered on time, delivered within budget and satisfies users' needs.

18. Which of the following is not a 'concern' during the management of a software project?
    A. Product quantity
    B. Time
    C. Project/Product information
    D. Money

Solution: (A). Product quantity would not include during the management of a software project

19. Black-box testing is also called _____.
    A. Specification-based testing
    B. Structural testing
    C. Verification
    D. Unit testing

Solution: (A). Black-box testing is another name for specification-based testing.

20. Alpha and Beta Testing are forms of
    A. Acceptance Testing
    B. Integration Testing

    C. System Testing
    D. Unit Testing

Solution: (A). Acceptance Testing

21. In which step of SDLC, actual programming of software code is done?
    A. Development and Documentation
    B. Maintenance and Evaluation
    C. Design
    D. Analysis

Solution: (A). The documentation explains the functions of the final product. The developer must discover adequate knowledge in the technical documentation to begin coding.

22. _____ is a strategy to achieve Software diversity.
    A. Explicit specification of different algorithms
    B. Different programming languages
    C. Different design methods and tools
    D. All of the mentioned

Solution: (D). Diversity refers to the ability to deliver the same functionality in a variety of ways so that essential components of a dependable system do not fail in the same way. Because we all have varied life experiences, backgrounds, and expertise, adding variety to the problem-solving process is essential because it allows us to come up with new ideas and methods.

23. _____ is an indirect measure of software development process.
    A. Cost
    B. Effort applied

    C. Efficiency
    D. All of the mentioned

Solution: (C). Efficiency is an indirect measure. Indirect measures also include products like maintainability, quality, functionality, complexity, reliability, and many more.

24. Which of the following is not an information domain required for determining function point in FPA?
    A. Number of user Input
    B. Number of user Inquiries
    C. Number of external Interfaces
    D. Number of errors

Solution: (D). Explanation: FPA includes five domains namely input, output, inquiries, interface and logical files

25. _____ is a measure of the degree of interdependence between modules.
    A. Cohesion
    B. Coupling
    C. None of the mentioned
    D. All of the mentioned

Solution: (B). Coupling or dependency is the degree to which each program module relies on each one of the other modules.

Answers:

1. A	2. C	3. C	4. B	5. A
6. B	7. C	8. C	9. D	10. A
11. C	12. C	13. A	14. B	15. D
16. B	17. D	18. A	19. A	20. A
21. A	22. D	23. C	24. D	25. B

9.8 PRACTICE QUESTION SETS

1. Who is the father of Software Engineering?
    A. Margaret Hamilton
    B. Watts S. Humphrey
    C. Alan Turing
    D. Boris Beizer
2. Which one of the following is NOT desired in a good Software Requirement Specifications (SRS) document?
    A. Functional Requirements
    B. Non-functional Requirements
    C. Goals of Implementation
    D. Algorithms for Software Implementation
3. _____ is a software development activity that is not a part of software processes.
    A. Validation
    B. Specification
    C. Development
    D. Dependence
4. The ISO quality assurance standard that applies to software Engineering is _____.
    A. ISO 9001 : 2000
    B. ISO 9000 : 2004
    C. ISO 9002 : 2001
    D. ISO 9003 : 2004
5. CASE stands for _____.
    A. Computer Aided Software Engineering
    B. Control Aided Science and Engineering

Solution: (B). Two modules are control coupled if one passes an element of control to another.

17. The aim of software engineering is to produce software that is _____.
    A. Fault-free
    B. Delivered on time
    C. Delivered within budget
    D. All of these

    Solution: (D). The aim of software engineering is to produce software that is fault-free, delivered on time, delivered within budget and satisfies users' needs.

18. Which of the following is not a 'concern' during the management of a software project?
    A. Product quantity
    B. Time
    C. Project/Product information
    D. Money

    Solution: (A). Product quantity would not include during the management of a software project

19. Black-box testing is also called _____.
    A. Specification-based testing
    B. Structural testing
    C. Verification
    D. Unit testing

    Solution: (A). Black-box testing is another name for specification-based testing.

20. Alpha and Beta Testing are forms of
    A. Acceptance Testing
    B. Integration Testing
    C. System Testing
    D. Unit Testing

    Solution: (A). Acceptance Testing

21. In which step of SDLC, actual programming of software code is done?
    A. Development and Documentation
    B. Maintenance and Evaluation
    C. Design
    D. Analysis

    Solution: (A). The documentation explains the functions of the final product. The developer must discover adequate knowledge in the technical documentation to begin coding.

22. _____ is a strategy to achieve Software diversity.
    A. Explicit specification of different algorithms
    B. Different programming languages
    C. Different design methods and tools
    D. All of the mentioned

    Solution: (D). Diversity refers to the ability to deliver the same functionality in a variety of ways so that essential components of a dependable system do not fail in the same way. Because we all have varied life experiences, backgrounds, and expertise, adding variety to the problem-solving process is essential because it allows us to come up with new ideas and methods.

23. _____ is an indirect measure of software development process.
    A. Cost
    B. Effort applied
    C. Efficiency
    D. All of the mentioned

    Solution: (C). Efficiency is an indirect measure. Indirect measures also include products like maintainability, quality, functionality, complexity, reliability, and many more.

24. Which of the following is not an information domain required for determining function point in FPA?
    A. Number of user Input
    B. Number of user Inquiries
    C. Number of external Interfaces
    D. Number of errors

    Solution: (D). Explanation: FPA includes five domains namely input, output, inquiries, interface and logical files

25. _____ is a measure of the degree of interdependence between modules.
    A. Cohesion
    B. Coupling
    C. None of the mentioned
    D. All of the mentioned

    Solution: (B). Coupling or dependency is the degree to which each program module relies on each one of the other modules.

Answers:

1. A	2. C	3. C	4. B	5. A
6. B	7. C	8. C	9. D	10. A
11. C	12. C	13. A	14. B	15. D
16. B	17. D	18. A	19. A	20. A
21. A	22. D	23. C	24. D	25. B

9.8 PRACTICE QUESTION SETS

1. Who is the father of Software Engineering?
   A. Margaret Hamilton
   B. Watts S. Humphrey
   C. Alan Turing
   D. Boris Beizer

2. Which one of the following is NOT desired in a good Software Requirement Specifications (SRS) document?
   A. Functional Requirements
   B. Non-functional Requirements
   C. Goals of Implementation
   D. Algorithms for Software Implementation

3. _____ is a software development activity that is not a part of software processes.
   A. Validation
   B. Specification
   C. Development
   D. Dependence

4. The ISO quality assurance standard that applies to software Engineering is _____.
   A. ISO 9001 : 2000
   B. ISO 9000 : 2004
   C. ISO 9002 : 2001
   D. ISO 9003 : 2004

5. CASE stands for _____.
   A. Computer Aided Software Engineering
   B. Control Aided Science and Engineering

C. Cost Aided System Experiments
D. None of the mentioned

6. What is a Functional Requirement?
A. Specifies the tasks the program must complete
B. Specifies the tasks the program should not complete
C. Specifies the tasks the program must not work
D. All of the mentioned

7. Why do bugs and failures occur in software?
A. Because of developers
B. Because of companies
C. Because of both companies and developers
D. None of the mentioned

8. Attributes of good software is _____.
A. Development
B. Maintainability and functionality
C. Functionality
D. Maintainability

9. What does SDLC stands for?
A. System Design Life Cycle
B. Software Design Life Cycle
C. Software Development Life Cycle
D. System Development Life Cycle

10. Who proposed the spiral model?
A. Barry Boehm
B. Pressman
C. Royce
D. IBM

11. _____ software development team has no permanent leader.
A. Controlled Centralized (CC)
B. Controlled Decentralized (CD)

C. Democratic Decentralized (DD)
D. None of the mentioned

12. What are agile manifesto principles?
A. Customer satisfaction
B. Face-to-face communication within a development team
C. Changes in requirements are welcome
D. All of the mentioned

13. Who proposed Function Points?
A. Albrecht
B. Jacobson
C. Boehm
D. Booch

14. _____ is a software development life cycle model that is chosen if the development team has less experience on similar projects.
A. Iterative Enhancement Model
B. RAD
C. Spiral
D. Waterfall

15. Which of the following testing methods uses fault simulation technique?
A. Unit testing
B. Beta testing
C. Mutation testing
D. Stress testing

16. _____ is not suitable for accommodating any change?
A. RAD Model
B. Waterfall Model
C. Build and Fix Model
D. Prototyping Model

17. The model which has a major disadvantage in terms of the coding

phase of a software life cycle model is _____.
A. Rad Model
B. Spiral Model
C. 4GT Model
D. Waterfall Model

18. Full form of the "COCOMO" model is _____.
A. Cost Constructive Estimation Model
B. Constructive Cost Estimation Model
C. Constructive Case Estimation Model
D. Constructive Cost Estimating Model

19. Which one of the following is not a software process quality?
A. Visibility
B. Timeliness
C. Productivity
D. Portability

20. _____ is an indirect measure of software development process.
A. Cost
B. Effort applied
C. Efficiency
D. All of the mentioned

21. Which of the following document contains the user system requirements?
A. SRD
B. DDD
C. SDD
D. SRS

22. _____ specification is also known as SRS document.
A. White-box

B. Grey-box
C. Black-box
D. None of the mentioned

23. The tools that support different stages of software development life cycle are called.....
A. CASE Tools
B. CAME tools
C. CAQE tool
D. CARE tools

24. Software Debugging is known as _____.
A. Identifying the task to be computerized
B. Creating program code
C. Creating the algorithm
D. Finding and correcting errors in the program code

25. RAD stands for _____.
A. Relative Application Development
B. Rapid Application Development
C. Rapid Application Document
D. Rapid and Design

26. Which one of the following models is not suitable for accommodating any change?
A. Build and Fix Model
B. Prototyping Model
C. RAD Model
D. Waterfall Model

27. Which is not one of the types of prototype of Prototyping Model?
A. Horizontal Prototype
B. Vertical Prototype

C. Diagonal Prototype
D. Domain Prototype

28. Which one of the following is not a phase of Prototyping Model?
A. Quick Design
B. Coding
C. Prototype Refinement
D. Engineer Product

29. RAD Model has _____.
A. 2 phases
B. 3 phase
C. 5 phases
D. 6 phases

30. What is the major drawback of using RAD Model?
A. Highly specialized and skilled developers/designers are required
B. Increases reusability of components
C. Encourages customer/client feedback
D. Increases reusability of components; Highly specialized and skilled developers/designers are required

31. Which model can be selected if user is involved in all the phases of SDLC?
A. Waterfall Model
B. Prototyping Model
C. RAD Model
D. Both Prototyping Model and RAD Model

32. The Incremental Model is a result of combination of elements of which two models?
A. Build and Fix Model, and Waterfall Model
B. Linear Model and RAD Model
C. Linear Model and Prototyping Model
D. Waterfall Model and RAD Model

33. Which two models does not allow defining requirements early in the cycle?
A. Waterfall and RAD
B. Prototyping and Spiral
C. Prototyping and RAD
D. Waterfall and Spiral

34. Which of the following life cycle model can be chosen if the development team has less experience on similar projects?
A. Spiral
B. Waterfall
C. RAD
D. Iterative Enhancement Model

35. Function points can be calculated by _____.
A. UFP * Cost
B. UFP * CAF
C. UFP * FAC
D. UFP * Productivity

36. Effective software project management focuses on the four P's. What are those four P's.
A. People, Performance, Payment, Product.
B. People, Product, Process, Project
C. People, Product, Performance, Project
D. All of the above

37. _____ and _____ are two kinds of software products.
A. CAD, CAM
B. Firmware, Embedded
C. Generic, Customized
D. None of the mentioned

38. Which one of the following is not an application of embedded software product?
A. Keypad control of a security system
B. Pattern recognition game playing
C. Digital function of dashboard display in a car
D. None of the mentioned

39. Purpose of process is to deliver software _____.
A. In time
B. With acceptable quality
C. That is cost-efficient
D. Both in time and with acceptable quality

40. Four types of change are encountered during the support phase. Which one of the following is not one that falls into such category?
A. Translation
B. Correction
C. Adaptation
D. Prevention

41. While estimating the cost of software, Lines of Code (LOC) and Function Points (FP) are used to measure which one of the following?
A. Length of code
B. Size of software
C. Functionality of software
D. None of the above

42. _____ is a process model that removes defects before they can precipitate serious hazards.
A. Incremental model
B. Spiral model
C. Clean room software engineering
D. Agile model

43. Equivalence partitioning is a _____ method that divides the input domain of a program into classes of data from which test cases can be derived.
A. White-box testing
B. Black-box testing
C. Orthogonal array testing
D. Stress testing

44. The incorrect activity among the following for the configuration management of a software system is _____.
A. Version management
B. System management
C. Change management
D. Internship management

45. SRS is also known as specification of _____.
A. White box testing
B. Integrated testing
C. Stress testing
D. Black-box testing

46. SRD stands for _____.
A. Software requirements definition
B. Structured requirements definition

C. Software requirements diagram
D. Structured requirements diagram

47. Software testing is _____.
A. The process of executing a program with the intent of finding errors
B. The process of establishing that errors are not present
C. The process of establishing confidence that a program does what it is supposed to do
D. The process of executing a program to show that it is working as per specifications

48. In terms of a system, finished products and information are examples of _____.
A. Input
B. Feedback
C. Control
D. Output

49. For a well-understood data processing application, it is best to use _____.
A. Prototyping model
B. Evolutionary model
C. Spiral model
D. Waterfall model

50. All the modules of the system are integrated and tested as complete system in the case of _____.
A. Bottom-up testing
B. Sandwich testing
C. Top-down testing
D. Big-Bang testing

Answers:

1. B	2. D	3. D	4. A	5. A
6. A	7. C	8. B	9. C	10. A
11. C	12. D	13. A	14. C	15. C
16. C	17. C	18. B	19. D	20. A
21. D	22. C	23. A	24. D	25. B
26. D	27. C	28. B	29. C	30. D
31. C	32. C	33. B	34. A	35. B
36. B	37. C	38. B	39. D	40. A
41. B	42. C	43. B	44. D	45. D
46. B	47. A	48. D	49. D	50. D

CHAPTER 10

MACHINE LEARNING

SYLLABUS

Classification, Decision Tree Learning, Artificial Neural Networks, Support Vector Machines, Bayesian Learning, Clustering, Hidden Markov Models.

10.1 INTRODUCTION

Machine learning is a growing technology which enables computers to learn automatically from past data. With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. It constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.

10.1.1 What is ML?

Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959.

10.1.2 ML Life Cycle

Machine learning life cycle involves seven major steps, which are given below:

Gathering Data, Data preparation, Data Wrangling, Analyse Data, Train the model, Test the model, Deployment.

10.1.3 ML in Current Scenario

ML has got a great advancement in its research, and it is present everywhere around us, such as self-driving cars, Amazon Alexa, Chatbots, recommender system, and many more. It includes

Supervised, unsupervised, and reinforcement learning with clustering, classification, decision tree, SVM algorithms, etc.

Modern machine learning models can be used for making various predictions, including weather prediction, disease prediction, stock market analysis, etc.

10.2 CLASSIFICATION

ML can be classified into three types:

(a) Supervised learning
- Classification
- Regression

(b) Unsupervised learning
- Clustering
- Association

(c) Reinforcement learning

10.2.1 Supervised Learning

Supervised learning algorithm take the data sample, *i.e.*, the training data and its associated output, *i.e.*, labels or responses with each data samples during the training process. It is called supervised because the whole process of learning can be thought as it is being supervised by a teacher or supervisor. The example of supervised learning is spam filtering. The main objective of supervised learning algorithms is to learn an association between input data samples and corresponding outputs after performing multiple training data instances.

For example, we have Input variables (X) and Output variable (Y), Now, apply an algorithm to learn the mapping function from the input to output as follows:

$Y = f(X)$

Now, the main objective would be to approximate the mapping function so well that even when we have new input data (X), we can easily predict the output variable (Y) for that new input data.

Supervised learning algorithms can be divided into following two broad classes.

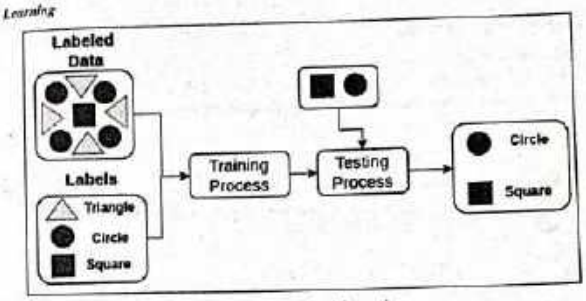


Fig. 10.1: Supervised Learning

10.2.1.1 Classification

The key objective of classification-based tasks is to predict categorial output labels or responses for the given input data. The output will be based on what the model has learned in training phase. As we know that the categorial output responses means unordered and discrete values, hence each output response will belong to a specific class or category. Some classification algorithms are:

- Random Forest
- Decision Trees
- Logistic Regression
- Support Vector Machines

10.2.1.2 Regression

The key objective of regression-based tasks is to predict output labels or responses which are continuous numeric values, for the given input data. The output will be based on what the model has learned in its training phase. Basically, regression models use the input data features (independent variables) and their corresponding continuous numeric output values (dependent or outcome variables) to learn specific association between inputs and corresponding outputs. Some regression algorithms are:

- Linear Regression
- Regression Trees
- Non-linear Regression
- Bayesian Linear Regression
- Polynomial Regression

10.2.2 Unsupervised Learning

Unsupervised learning is a type of ML in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

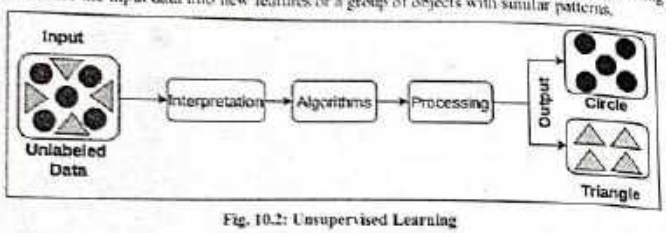


Fig. 10.2: Unsupervised Learning

10.2.2.1 Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

10.2.2.2 Association

Association is used to analyze large dataset to find patterns which further represents the interesting relationships between various items. It is also termed as Association Rule Mining or Market basket analysis which is mainly used to analyze customer shopping patterns.

10.2.3 Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points. Hence, it improves its performance.

10.3 DECISION TREE LEARNING

Decision Tree is a supervised learning algorithm that can be used for both classification and Regression problems, but mostly it is preferred for solving classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub-trees.

Below diagram explains the general structure of a decision tree:

10.3.1 Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

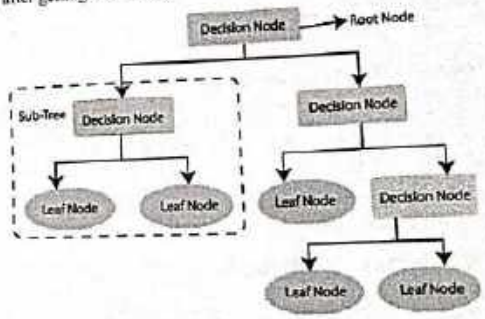


Fig. 10.3: Decision Tree Structure

- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub-tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child Node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

10.3.2 Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**

By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain:** Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

Information Gain = Entropy (S) − [(Weighted Avg) * Entropy (each feature)]

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data.

Entropy $(S) = -P(Y)\log_2 P(Y) - P(N)\log_2 P(N)$

where, S = Total number of samples; $P(Y)$ = probability of yes; $P(N)$= probability of no.

- **Gini Index:** Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits. Gini index can be calculated using the below formula:

$$Gini\ Index = 1 - \sum_i P_i^2$$

10.3.3 Pruning

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree. A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used:

- Cost Complexity Pruning
- Reduced Error Pruning.

10.4 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) are a set of algorithms that tries to recognize the patterns, relationships, and information from the data through the process which is inspired by and works like the human brain/biology.

10.4.1 Architecture of an ANN

Artificial Neural Network primarily consists of three layers:

- **Input Layer:** It accepts inputs in several different formats provided by the programmer.

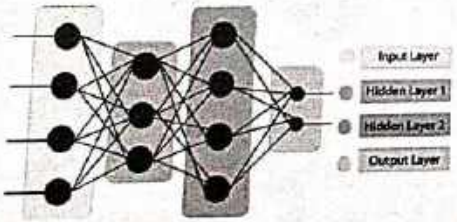


Fig. 10.4: Structure of ANN

- **Hidden Layer:** It is present in-between input and output layers. It performs all the calculations to find hidden features and patterns.
- **Output Layer:** The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} W_i * X_i + b$$

It determines how the weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

10.4.2 Types of ANN

- **Feedback ANN:** Here, the output returns into the network to accomplish the best-evolved results internally.
- **Feed-forward ANN:** A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least, one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behaviour of the associated neurons, and the output is decided.

10.5 SUPPORT VECTOR MACHINES

Support Vector Machine or SVM is one of the most popular supervised learning algorithms, which is used for classification as well as regression problems. However, primarily, it is used for classification problems in machine learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a **hyperplane.**

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane: the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called **support vectors.** The distance between the vectors and the hyperplane is called as **margin.** And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the **optimal hyperplane.**

SVM algorithm can be used for Face detection, image classification, text categorization, etc.

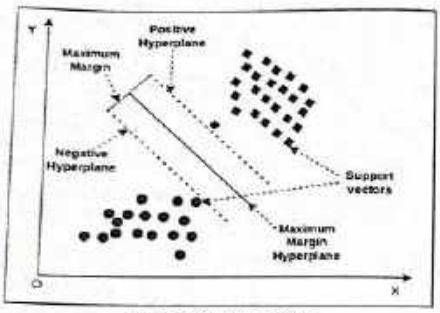


Fig. 10.5: Structure of SVM

10.5.1 Types of SVM

SVM can be divided into two categories:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

10.6 BAYESIAN LEARNING

Naïve Bayes' algorithm is a supervised learning algorithm, which is based on Bayes' theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes' Classifier is one of the simple and most effective classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes' Algorithm are spam filtration, Sentimental analysis, and classifying articles.

10.6.1 Bayes' Theorem

Bayes' theorem is also known as **Bayes' Rule** or **Bayes' Law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:

$$P(A|B) = P(B|A)\,P(A) / P(B)$$

- $P(A|B)$ **is Posterior Probability:** Probability of hypothesis A on the observed event B.
- $P(B|A)$ **is Likelihood Probability:** Probability of the evidence given that the probability of a hypothesis is true.
- $P(A)$ **is Prior Probability:** Probability of hypothesis before observing the evidence.
- $P(B)$ **is Marginal Probability:** Probability of evidence.

10.6.2 Types of Naive Bayes' Model

There are three types of Naive Bayes Model:

- **Gaussian:** The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial:** The Multinomial Naïve Bayes' classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, Education, etc.

  The classifier uses the frequency of words for the predictors.
- **Bernoulli:** The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Boolean variables. Such as if a particular word is present or not in a document. This model is also famous for document classification task.

10.6.3 Applications of Naive Bayes' Algorithm

- It is used for credit scoring.
- It is used in medical data classification.
- It can be used in real-time predictions because Naïve Bayes' Classifier is an eager learner.
- It is used in text classification such as Spam filtering and Sentiment analysis.

10.7 CLUSTERING

Clustering is a unsupervised learning technique where data is grouped based on the similarity of the data points. The basic principle behind cluster is the assignment of a given set of observations into subgroups or clusters such that observations present in the same cluster possess a degree of similarity. Clustering is important because it determines the intrinsic grouping among the present unlabeled data. They basically make some assumptions about data points to constitute their similarity. Each assumption will construct different but equally valid clusters.

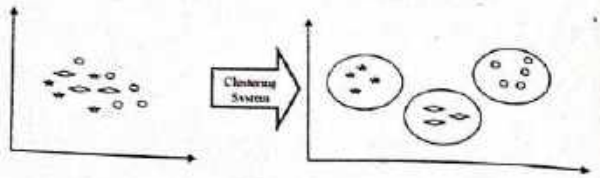


Fig. 10.6: Clustering

10.7.1 Types of Clustering Algorithms

There are five distinct types of clustering algorithms:

- **Partitioning-based Clustering:** It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the centroid-based method. The most common example of partitioning clustering is the K-Means Clustering algorithm.
- **Hierarchical Clustering:** Here, the dataset is divided into clusters to create a tree-like structure, which is also called a dendrogram. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the **Agglomerative Hierarchical algorithm**.
- **Distribution Model-based Clustering:** Here, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly Gaussian Distribution. For example, **Expectation-Maximization Clustering algorithm** that uses Gaussian Mixture Models (GMM).
- **Density-based Clustering:** Here, the clusters are formed as the dense region. The advantage of these methods is that they have good accuracy as well as good ability to merge two clusters, *e.g.*, Density-based Spatial Clustering of Applications with Noise (DBSCAN), Ordering Points to Identify Clustering Structure (OPTICS), etc.
- **Fuzzy Clustering:** Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster, *e.g.*, **Fuzzy C-means** algorithm.

10.7.2 Applications of Clustering

- **Data summarization and compression:** Collaborative systems and Customer segmentation
- **Serve as a key intermediate step for other data mining tasks:** Trend detection in dynamic data
- **Social network analysis:** Generating sequences in images, videos or audios
- **Biological data analysis:** Identification of Cancer cells, etc.

10.8 HIDDEN MARKOV MODELS

Hidden Markov Model is an temporal probabilistic model for which a single discontinuous random variable determines all the states of the system. It means that, Possible values of variable = Possible states in the system. For example, sunlight can be the variable and sun can be the only possible state. The structure of Hidden Markov Model is restricted to the fact that basic algorithms can be implemented using matrix representations. HMM provides solution of three problems: evaluation, decoding and learning to find most likelihood classification.

10.8.1 Concept

In Hidden Markov Model, every individual states has limited number of transitions and emissions. Probability is assigned for each transition between states. Hence, the past states are totally independent of future states. The fact that HMM is called hidden because of its ability of being a memory less process, *i.e.*, its future and past states are not dependent on each other. Since HMM is rich in mathematical structure, it can be implemented for practical applications. This can be achieved on two algorithms called as:

(a) Forward Algorithm
(b) Backward Algorithm.

10.8.2 Applications

Some important application areas are:

(a) Speech Recognition.
(b) Gesture Recognition.
(c) Language Recognition.
(d) Motion Sensing and Analysis.
(e) Protein Folding.

10.9 SOLVED QUESTIONS

1. What is Machine Learning?
   - Artificial Intelligence
   - Deep Learning
   - Data Statistics

   A. Only (i)
   B. (i) and (ii)
   C. All
   D. None

   Solution: (B). Data statistics is not a machine learning process.

2. Which of the following is not type of learning?

   A. Unsupervised Learning
   B. Supervised Learning
   C. Semi-unsupervised Learning
   D. Reinforcement Learning

   Solution: (C). Basically, there are three types of learning – Supervised, Unsupervised and Reinforcement learning.

3. Real-time decisions, Game AI, Learning Tasks, Skill Acquisition and

Robot Navigation are THE applications of which of the following:

A. Supervised Learning: Classification

B. Reinforcement Learning

C. Unsupervised Learning: Clustering

D. Unsupervised Learning: Regression

Solution: (B). All of the above cases we have to learn from the past experience.

4. Targeted Marketing, Recommended Systems and Customer Segmentation are the applications in which of the following:

A. Supervised Learning: Classification

B. Unsupervised Learning: Clustering

C. Unsupervised Learning: Regression

D. Reinforcement Learning

Solution: (B). All of above mentioned systems are Clustering system.

5. Fraud Detection, Image Classification, Diagnostic, and Customer Retention are applications in which of the following:

A. Unsupervised Learning: Regression

B. Supervised Learning: Classification

C. Unsupervised Learning: Clustering

D. Reinforcement Learning

Solution: (B). All of the above mentioned systems are classification system.

6. FIND-S algorithm ignores _____.

A. Negative

B. Positive

C. Both

D. None of the above

Solution: (A). FIND-S algorithm only considers positive.

7. Which is true for neural networks?

A. Each node computes it's weighted input

B. Node could be in excited state or non-excited state

C. It has set of nodes and connections

D. All of the above

Solution: (D). All the given options are related to neural networks.

8. What is true regarding back propagation rule?

A. Error in output is propagated backwards only to determine weight updates

B. There is no feedback of signal at any stage

C. It is also called generalized delta rule

D. All of the above

Solution: (D). All are true for back propagation algorithm.

9. A 3-input neuron has weights 1, 4 and 3. The transfer function is linear with the constant of proportionality being equal to 3. The inputs are 4, 8 and 5 respectively. What will be the output?

A. 139

B. 153

C. 162

D. 169

Solution: (B). The output is found by multiplying the weights with their respective inputs, summing the results and multiplying with the transfer function. Therefore, Output = 3 * (1 * 4 + 4 * 8 + 3 * 5) = 153.

10. How many types of agents are there in artificial intelligence?

A. 1

B. 3

C. 4

D. 2

Solution: (C). The four types of agents are Simple reflex, Model-based, Goal-based and Utility-based agents.

11. Neural Networks are complex _____ functions with many parameters.

A. Linear

B. Non-linear

C. Discrete

D. Exponential

Solution: (A). Linear function.

12. _____ terms are required for building a Bayes' model.

A. 1

B. 2

C. 3

D. 4

Solution: (C). The three required terms are a conditional probability and two unconditional probability.

13. Examples of Naïve Bayes' Algorithm is/are _____.

A. Spam filtration

B. Sentimental analysis

C. Classifying articles

D. All of the above

Solution: (D). All of the above mentioned options are Naïve Bayes' Algorithm.

14. Naïve Bayes' algorithm is based on _____ and used for solving classification problems.

A. Bayes' Theorem

B. Candidate elimination algorithm

C. EM algorithm

D. None of the above

Solution: (A). Naïve Bayes' algorithm are based on Bayes' Theorem.

15. Full form of MDL?

A. Minimum Description Length

B. Maximum Description Length

C. Minimum Domain Length

D. None of these

Solution: (B). Maximum Description Length.

16. PAC stands for _____.

A. Probably Approximate Correct

B. Probably Approx Correct

C. Probably Approximate Computation

D. Probably Approx Computation

Solution: (A). Probably Approximate Correct.

17. What are the advantages of Nearest neighbour algo?

A. Training is very fast
B. Can learn complex target functions
C. Don't lose information
D. All of these

Solution: (D). All of these.

18. What are the difficulties with K-nearest neighbour algo?
A. Calculate the distance of the test case from all training cases
B. Curse of dimensionality
C. Both A and B
D. None of these

Solution: (C).

19. What is/are true about Distance-weighted KNN?
A. The weight of the neighbour is considered
B. The distance of the neighbour is considered
C. Both A and B
D. None of these

Solution: (C). In Distance-weighted KNN, both distance and weights are considered.

20. It was shown that the Naive Bayesian method _____.
A. Can be much more accurate than the optimal Bayesian method
B. Is always worse off than the optimal Bayesian method
C. Can be almost optimal only when attributes are independent
D. Can be almost optimal when some attributes are dependent

Solution: (C).

21. Example of Reinforcement learning:
A. Chess game
B. Object recognition
C. Weather conditions
D. Price of house

Solution: (A). Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance

22. What is overfitting?
A. When a predictive model is accurate but takes too long to run
B. When the model learns specifics of the training data that cannot be generalized to a larger dataset
C. When you perform hyper-parameter tuning and performance degrades
D. When you apply a powerful deep learning algorithm to a simple machine learning problem

Solution: (C). Overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points.

23. In a Decision Tree, Leaf Node represents _____.
A. One of the class label
B. One of the complete observation
C. One of the attribute
D. None of the mentioned

Solution: (A).

24. _____ is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (Y).
A. Regression
B. Clustering
C. Decision Tree
D. Classification

Solution: (A).

25. Agglomerative clustering follows _____.
A. Top-down approach
B. Bottom-up approach
C. Partitional clustering
D. Model-based clustering

Solution: (B).

Answers:

1. B	2. C	3. B	4. B	5. B
6. A	7. D	8. D	9. B	10. C
11. A	12. C	13. D	14. A	15. B
16. A	17. D	18. C	19. C	20. C
21. A	22. C	23. A	24. A	25. B

10.10 PRACTICE QUESTION SETS

1. What is Machine Learning (ML)?
A. The autonomous acquisition of knowledge through the use of manual programs
B. The selective acquisition of knowledge through the use of computer programs
C. The selective acquisition of knowledge through the use of manual programs
D. The autonomous acquisition of knowledge through the use of computer programs

2. Father of Machine Learning (ML):
A. Geoffrey Chaucer
B. Geoffrey Hill
C. Geoffrey Everest Hinton
D. None of the above

3. Which is FALSE regarding regression?
A. It may be used for interpretation
B. It is used for prediction
C. It discovers causal relationships
D. It relates inputs to outputs

4. Choose the correct option regarding machine learning (ML) and artificial intelligence (AI):
A. ML is a set of techniques that turns a dataset into a software
B. AI is a software that can emulate the human mind
C. ML is an alternate way of programming intelligent machines
D. All of the above

5. Which of the factors affect the performance of the learner system does not include?
A. Good data structures
B. Representation scheme used
C. Training scenario
D. Type of feedback

6. Which of the following does not include different learning methods?
A. Analogy
B. Introduction
C. Memorization
D. Deduction

7. In language understanding, the levels of knowledge that does not include?

A. Empirical
B. Logical
C. Phonological
D. Syntactic

8. Designing a machine learning approach involves _____.
A. Choosing the type of training experience
B. Choosing the target function to be learned
C. Choosing a function approximation algorithm
D. All of the above

9. Concept learning inferred a _____ valued function from training examples of its input and output.
A. Decimal
B. Hexadecimal
C. Boolean
D. All of the above

10. Which of the following is not a supervised learning?
A. Naive Bayesian
B. PCA
C. Linear Regression
D. Decision Tree

11. What kind of learning algorithm is used for "facial identities or facial expressions"?
A. Prediction
B. Recognition Patterns
C. Generating Patterns
D. Recognizing Anomalies

12. Which of the following is not function of symbolic in the various function representation of ML?
A. Rules in Proportional Logic
B. Hidden-Markov Models (HMM)
C. Rules in First-order Predicate Logic
D. Decision Trees

13. Which of the following is not numerical functions in the various function representation of ML?
A. Neural Network
B. Support Vector Machines
C. Case-based
D. Linear Regression

14. FIND-S Algorithm starts from the most specific hypothesis and generalize it by considering only _____.
A. Negative
B. Positive
C. Negative or Positive
D. None of the above

15. The Candidate-Elimination Algorithm represents the _____.
A. Solution Space
B. Version Space
C. Elimination Space
D. All of the above

16. Inductive learning takes examples and generalizes rather than starting with _____.
A. Inductive
B. Existing
C. Deductive
D. None of these

17. Which of the following is a widely used and effective ML algorithm based on the idea of bagging?
A. Decision Tree
B. Random Forest
C. Regression
D. Classification

18. Which of the following is a disadvantage of decision trees?
A. Decision trees are prone to be overfit
B. Decision trees are robust to outliers
C. Factor analysis
D. None of the above

19. What is perceptron?
A. A single layer feed-forward neural network with pre-processing
B. A neural network that contains feedback
C. A double layer auto-associative neural network
D. An auto-associative neural network

20. What is back propagation?
A. It is another name given to the curvy function in the perceptron
B. It is the transmission of error back through the network to allow weights to be adjusted so that the network can learn
C. It is another name given to the curvy function in the perceptron
D. None of the above

21. What is the objective of back propagation algorithm?

A. To develop learning algorithm for multi-layer feed-forward neural network, so that network can be trained to capture the mapping implicitly
B. To develop learning algorithm for multi-layer feed-forward neural network
C. To develop learning algorithm for single layer feed-forward neural network
D. All of the above

22. An auto-associative network is _____.
A. A neural network that has only one loop
B. A neural network that contains feedback
C. A single layer feed-forward neural network with pre-processing
D. A neural network that contains no loops

23. What of the following is true regarding back propagation rule?
A. Hidden layers output is not at all important, they are only meant for supporting input and output layers
B. Actual output is determined by computing the outputs of units for each hidden layer
C. It is a feedback neural network
D. None of the above

24. The general limitations of back propagation rule is/are _____.
A. Scaling
B. Slow convergence
C. Local minima problem
D. All of the above

25. What is the meaning of generalized in statement "back propagation is a generalized delta rule"?
    A. Because delta is applied to only input and output layers, thus making it more simple and generalized
    B. It has no significance
    C. Because delta rule can be extended to hidden layer units
    D. None of the above
26. The general tasks that are performed with back propagation algorithm:
    A. Pattern mapping
    B. Prediction
    C. Function approximation
    D. All of the above
27. The network that involves backward links from output to the input and hidden layers is known as _____.
    A. Recurrent neural network
    B. Self-organizing maps
    C. Perceptrons
    D. Single layered perceptron
28. Which of the following is/are the decision tree nodes?
    A. End Nodes
    B. Decision Nodes
    C. Chance Nodes
    D. All of the above
29. End nodes are represented by which of the following:
    A. Solar street light
    B. Triangles
    C. Circles
    D. Squares
30. Decision Nodes are represented by which of the following:
    A. Solar street light
    B. Triangles
    C. Circles
    D. Squares
31. Chance Nodes are represented by which of the following:
    A. Solar street light
    B. Triangles
    C. Circles
    D. Squares
32. _____ provides ways and means of weighing up the desirability of goals and the likelihood of achieving.
    A. Utility theory
    B. Decision theory
    C. Bayesian networks
    D. Probability theory
33. Which of the following is correct about the Naive Bayes?
    A. Assumes that all the features in a dataset are independent
    B. Assumes that all the features in a dataset are equally important
    C. Both
    D. All of the above
34. Naïve Bayes' Algorithm is a _____ learning algorithm.
    A. Supervised
    B. Reinforcement
    C. Unsupervised
    D. None of these
35. Types of Naïve Bayes' Model:
    A. Gaussian
    B. Multinomial

    C. Bernoulli
    D. All of the above
36. Disadvantages of Naïve Bayes' Classifier:
    A. Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between
    B. It performs well in multi-class predictions as compared to the other
    C. Naive Bayes is one of the fast and easy ML algorithms to predict a class of datasets
    D. It is the most popular choice for text classification problems
37. The benefit of Naive Bayes:
    A. Naive Bayes is one of the fast and easy ML algorithms to predict a class of datasets
    B. It is the most popular choice for text classification problems
    C. It can be used for binary as well as multi-class
    D. All of the above
38. What is/are advantage(s) of Distance-weighted k-NN over k-NN?
    A. Robust to noisy training data
    B. Quite effective when a sufficient large set of training data is provided
    C. Both A and B
    D. None of these
39. How many types of layer in radial basis function neural networks?
    A. 3
    B. 2
    C. 1
    D. 4
40. The neurons in the hidden layer contains Gaussian transfer function whose output are _____ to the distance from the centre of the neuron.
    A. Directly
    B. Inversely
    C. Equal
    D. None of these
41. In k-NN algorithm, given a set of training examples and the value of $k <$ size of training set (n), the algorithm predicts the class of a test example to be the. What is/are advantages of CBR?
    A. Least frequent class among the classes of k closest training
    B. Most frequent class among the classes of k closest training
    C. Class of the closest
    D. Most frequent class among the classes of the k farthest training examples
42. Features of Reinforcement learning:
    A. Set of problem rather than set of techniques
    B. RL is training by reward and
    C. RL is learning from trial and error with the
    D. All of these
43. Which type of feedback used by RL?
    A. Purely Instructive feedback
    B. Purely Evaluative feedback
    C. Both A and B
    D. None of these
44. What is/are the problem solving methods for RL?

A. Dynamic programming
B. Monte Carlo Methods
C. Temporal-difference learning
D. All of these

45. Which modifies the performance element so that it makes better decision?
A. Performance element
B. Changing element
C. Learning element
D. None of the mentioned

46. How the decision tree reaches its decision?
A. Single test
B. Two test
C. Sequence of test
D. No test

47. Which of the following is a disadvantage of decision trees?
A. Factor analysis
B. Decision trees are robust to outliers
C. Decision trees are prone to be overfit
D. None of the above

48. Which of the following are the advantage/s of Decision Trees?
A. Possible Scenarios can be added
B. Use a white box model, if given result is provided by a model
C. Worst, best and expected values can be determined for different scenarios
D. All of the mentioned

49. What is the mathematical likelihood that something will occur?
A. Classification
B. Probability
C. Naïve Bayes' Classifier
D. None of the other

50. What does the Bayesian network provides?
A. Complete description of the domain
B. Partial description of the domain
C. Complete description of the problem
D. None of the mentioned

Answers:

1. D	2. C	3. C	4. D	5. A
6. B	7. A	8. D	9. C	10. B
11. B	12. B	13. C	14. B	15. B
16. B	17. B	18. A	19. A	20. B
21. A	22. B	23. B	24. D	25. C
26. D	27. A	28. D	29. B	30. D
31. C	32. A	33. C	34. A	35. D
36. A	37. D	38. C	39. A	40. B
41. B	42. D	43. B	44. D	45. C
46. C	47. C	48. D	49. C	50. C