Computer Applications for ICSE Class X

1. Concepts of Objects:

1. What is meant by OOP concept? Object oriented programming means that more importance is give to the objects rather than to the procedure to solve a problem. It involves creation of objects that are very similar to real world objects.

2. Define object? Object is an identifiable entity with some characteristics and behaviour. Example: CAR→ Characteristics are wheels, steering, brakes, clutch etc. Behaviour → movement, control functions etc.

3. What are the principals of OOP? Abstraction, Encapsulation, Inheritance, Data Hiding etc.

4. Define Abstraction? Abstraction refers to the act of representing essential features without including the background details or explanations. Exp: While driving a car we only use steering, accelerator, brakes etc. and are not worried about the internal mechanism.

5. Define Encapsulation? Also termed as Data Hiding, is a way of combining data and function (that operates on that data) under a single unit (class). Thus encapsulation hides the data from the outside world and does not allows direct access of data. Data can be only accessed by functions.

6. What is Data Hiding? Data hiding means that the actual working of an object is hidden from the outside world. We are able to drive a CAR without knowing much of its internal details that are encapsulated, and can be only used through some functions.

7. What is a Method? It is a function, which represents the specific behavior of an object.

8. What is a Message? How do objects interact with each other? Message is a way of sending and receiving information to and from another object. When the object needs to interact with one another, they pass information to one another known as **message passing.**

9. How is Encapsulation related to Abstraction? Encapsulation is a way to implement data abstraction, by hiding the details of implementation of an object.

2. Introducing Classes:

1. Define Class? Class represents a set of objects that share common characteristics and behaviour. Class is an object factory. Exp: A Class of Cell Phone have many objects like: Nokia, Samsung, Motorola etc.

2. How are objects related to a class? Objects are instances of a class. The object represents the abstraction represented by the class in the real sense.

3. What is object factory? Why is class termed as object factory? An object factory is a producer of objects. It has information regarding the type, characteristics and behavior of an object.

Class is termed as an object factory because it contains all the statements needed to create an object and the operations that the object will be able to perform.

3(a) What is Modularity? The act of partitioning a program into individual components is called Modularity. 1. It reduces complexity 2. Creates a well defined documented boundaries within the program

4. Define Inheritance? Inheritance is the process by which one object acquires the properties of another object. It also allows creation of new class from an existing class and adds properties to the new class. *Example:* Alsatian is a part of class DOG, dog is a part of class MAMMAL and mammal comes under the class ANIMAL. Here Alsatian inherits the characteristics of dog, dog inherits the characteristics of mammal and mammal inherits the characteristics of animal class. The main class whose properties are inherited by another class is known as the Base class, Parent class or the Super class. The inherited class is known as the sub class, child class or the derived class.

5. Define Polymorphism? Polymorphism allows two or more classes or function to respond to the same message in different ways. It is a process where the same message can be processed in different ways depending on the type of data they act upon. *Example:* If a message of adding two values is given to two different functions having different data types they will produce different results. ADD integer values 5 + 4 = 9. ADD string values "5" + "4" = "54". Polymorphism is also termed as function overloading.

6. What is instance variable and class variable? Instance variable is an object that contains the information about the class variables and class functions. It is an instance of class. Class variables are individual variables of different types. Their behaviour is represented by the functions of that class.

7. What are Global variables? Variables declared within the scope of the class are global variables. They are of two types Static and Instance (nonstatic/Class) variables.

Static Variable: These are the data members that are declared once for the class. All object of the class type share these data members as there is a single copy of these variables available in the memory. Keyword *'static'* in the variable declaration makes the variable static.

Instance Variables: These are the data members that are created for every object of the class.

3. Introducing JAVA

1. Explain – "Java is both a programming language and a platform"? As a programming language we can create many computer applications. As a platform java is designed to run highly interactive, dynamic and secure applications for computers in a network.

2. Who developed Java? What is the initial name of Java? *James Gosling in 1991 developed* ***Oak (Java) language*** for programming intelligent consumer electronic devices.

3. What do you understand by Byte Code or Java Byte Code? After compilation Java does not produces *native executable code* for a particular machine but it produce a special format called **byte code**. It uses a 2-byte character code called ***Unicode,*** which is platformindependent i.e. recognized by all types of platforms whether dos, windows, Linux, Unix etc. A java **Byte Code** is a machine instruction for Java processor chip called Java Virtual Machine (JVM). Unicode supports 128 characters same as common ASCII character set, next 128 characters same as upper 128 characters of ISO Latin-1 which are extended ASCII character set. Other are 65280 characters.

4. What is source code? A program written by a programmer in ***high level language*** (English language) called the **source code.**

5. What is machine language code? Is a language which the computer can understand? After compiling the source code we get the machine code for that program which is understood by the computer.
6. What is High Level Language & Low Level Language? High Level Language is simple English language in which the program is written which is then converted by the language processors into Low Level Language known as machine code.
7. Define compilation? The process of converting source code to machine code is called compilation.
8. Define compiler and Interpreter? Compilation is done by language processors like compilers (COBOL) and interpreters (Basic). Compilers convert the whole program in one go from HLL to LLL, so compilation is fast but debugging (finding and rectification of errors) is difficult as it displays a list of errors. Interpreters convert the program from HLL to LLL line-by-line, so compilation is slow, debugging is easy as errors are detected and corrected one by one.
9. What do you understand by *Native Executive Code*? The machine code produced after compilation is called **native executable code.** It depends a lot on the platform it is executed upon. Thus different platforms are required to run different machine codes.
10. Explain - Java interpreter or Java Virtual Machine (JVM)? A java **Byte Code** is a machine instruction for Java processor chip called Java Virtual Machine (JVM). Java interpreter running on any machine appears and behaves like a "virtual" processor chip. [Virtual – appears real though unreal, gives a feel of being real.]
11. Give characteristics (features) of Java? Write Once Run Anywhere (WORA) Light weight
code – no huge coding is required even for large programs.
Security – It offers many security features.
Build in Graphics - features are offered by java.
Object Oriented Language – so very near to real world
Supports Multimedia – java is suitable to process video, audio, animation and graphics in Internet environment.
Platform Independent – change of platform does not affect the original java program or application.
Open product - freely available to all.
12. Explain "Write Once Run Anywhere" (WORA) feature of Java? This means that a program once written in Java Language can be easily executed and run on any platform.
13. What is a comment? Comments are English statement written to explain the code / logic of the program. The computer ignores them during compilation. Comments help in understanding the logic of program and documenting the program. The different ways of giving the comments are:
// is used for giving *single line comment,*
/* */ is used for giving *multi line comment*. Comment can be of more than two lines.
/** */ is used for documentation
14. Name the types of Java Programs? Java Internet Applets and Stand Alone applications.

4. Class as Basis of All Computation

1. What character set is used by Java? Java uses a 2-byte character code called ***Unicode,*** which is platform-independent i.e. recognized by all types of platforms whether dos, windows, Linux, Unix etc.

2. Define Tokens? Tokens are smallest individual unit of a program. Exp: Keywords, Identifiers, Literals etc.

3. Define Keywords? They are the reserved words used for special purpose in the language. They convey a special meaning to the compiler. They can't be used as identifiers. Exp: for, int, case, if, void, class, do etc.

4. Explain –"All keywords are reserve words, but all reserve words are not keywords".

Why *true, false and null* are termed as reserve words and not keywords? Can they be used as variables? Name reserve words that are not keywords? *True, false and null* are termed as reserve words as they cannot be used as identifiers or variables but they are not keywords as they do not convey a special meaning to the language compiler. Thus we can say that all keywords are reserve words but all reserve words are not keywords as all reserve words are not meant for some special purpose.

5. Define identifiers? Identifiers are fundamental building blocks of a program and are used to name different parts of the program like name of variable, objects, classes, functions, arrays etc.

6. Give characteristics of valid identifiers? A valid identifier must not begin with a digit, and must not contain any special character or space. Identifiers can have alphabets, digits, $ sign and underscore. They must not be a keyword or reserve word. They can be of any length and are case sensitive.

7. Are upper case and lower case identifiers treated similar in Java? No, they are not treated similar. Java is case sensitive as it treats upper and lower case letters differently.

8. Define literals? Literals are constants. They are fixed data values that remain fixed during program execution. Exp: Integer literals (5,6,8,-56,34 etc), floating literals (5.0,4.7, -7.5, 0.5 etc.), Boolean literals (true, false), the null literal (null), character literal ('k', 'r', 'a', '6', '9', '#', '@' etc.) any one character enclosed within single quotes and string literals ("Alok", "Delhi", "India", "786" etc.) any group of characters enclosed within double quotes.

9. What are Boolean and null literals? A Boolean literals are reserve constant words *true* and *false*. A Boolean literal can be either true or false. A null literal is always of the null type. It has one value, the null reference, represented by the constant *null*.

10. Give difference between character and string literal? ['a' and "a"]? A character literal is any alphabet, digit or special character enclosed within *single quotes*. Its length is always ONE. It is recognised by its ASCII code by the compiler. String literals are a group of characters (multiple character) enclosed within *double quotes*. Its length is equal to the number of characters enclosed within double quotes.

11. Explain Non-graphic characters or escape sequence characters? Non-graphic characters or escape sequence characters are those character constants that cannot be typed directly from keyboard e.g., backspace, tabs etc. They are represented by escape sequences by using a backslash (\) followed by one character. Exp: \t, \a, \n, \f etc. They are placed within double quotes with the print statement for use.

12. Explain any four Escape sequence characters? \t → for horizontal tab and provides eight spaces, \a→ gives a beep sound, \n→ gives a new line and transfers the control to the next line, \0→ used as Null.

13. What are separators? Separators are special characters that are used to group the items in a program code. They tell the compiler that these grouped items belong to one particular code. Example: **P*arentheses* ( )** defines and contains list of parameters for a function or constructor.

Also used in an expression so that the expressions within parentheses are solved first. They are also used in control structures like for, while, if. C**urly braces** { } forms a block and defines the scope of a class, function, for, while, do-while, if, switch and may contain executable statements. **Square brackets** [ ] are used to declare array, define an array type and its size and is used as the subscript for array. **Semicolon;** is used to terminate a statement. **Comma,** is used as a separator when many variables are declared by the same type. Also used as separator in list of parameters of function. **Dot .** is used to access the data members and member functions of a class.

14. What are operators? Operators represent operations or specific tasks. Arithmetic (+ - * / %), relational (< <= > >= ==), increment (++) decrement (- -), logical (&& || !), ternary (? :) operators.

15. What do Data types mean? Data types are means to identify the type of data. Exp: int, float, char, double.

16. Explain the two types of Data Types? Primitive data types that come as apart of the language. They store a give type of value: byte, short, int, long, float, double, char, Boolean. **Reference data types** are constructed from primitive data types: classes, arrays and interfaces. Exp: int arr[ ]= new int [10]; A reference data type is used to store the memory address of an object.

17. How are primitive and reference data types related? Primitive data types are fundamental components that come as a part of the language and reference data types are constructed from primitive data types.

18. Name and give the size occupied by Integer Primitive Data types?

19. Name and give the size occupied by Fractional Primitive Data types?

20. Give different suffixes to specify data types of a data value? L/l for long, F/f for float, D/d for double.

21. Name and give the size occupied by Character Data type?

22. What do reference data types mean? What is rvalue (actual read-value) and lvalue (location value) of variable? Reference data types are data elements whose value is an address. Instead of holding the real-value (rvalue) they hold the location value (el-value / lvalue) of the variable. Example: int a = 50 where 'a' is a variable at memory location address say x021. Here rvalue of variable 'a' is 50 and the lvalue is x021. Reference data types referrer any variable by there lvalue thus a variable is now referred by a different name. Thus if the value of reference variable changes the value of actual variable also changes.

23. Define variable? A variable is a name give to a memory location to store a value of a given data type.

24. How do we declare a variable? Syntax → datatype variable Name; (int age; float marks, Boolean true;)

25. What is meant by variable scope? Scope of variable is that program region within which a variable is accessible. A variable is accessible within the set of braces it is declared.

26. What is local and global variable? Global variable is a variable that is accessible by all the member functions of a class. Local variable is declared locally and its accessibility is limited to the function scope or block { } where it is declared. Example: class xyz { int a, b; void main( ) { int c; ….. …. } } Here a and b are global variables of class xyz and c is local variable of function main.

27. How do we initialise a variable? Initialisation of variable means giving the initial value to a variable. A variable can be initialised after declaration → int age; age=18; A variable can be initialised during declaration→ int age=18, of can be initialised dynamically.

28. Define and explain Dynamic initialisation of a variable? A variable is said to be initialized dynamically when it is initialised by the return value of expression or function. Example: int c=a+b; or int c=Math.sqrt(a); Here in both the examples c has been initialised dynamically.

29. Define constants? Give advantage of constant variables? Constants are those variables whose value is not going to change during program execution. A constant variable can be declared by using a keywords – final. As→ final double taxRate=0.25; Constants make your program easier to read and check for correctness. If a constant needs to be changed only the declaration is changed at one place.

30. Define operands & Expression? Operands are termed as objects of the operations or specific tasks. Operands are variables used with the expressions. **Expression** is a valid arithmetic expression having a combination of arithmetic operators and variable that results into a value.

31. What are Arithmetic operators? Give difference between / and % arithmetic operators? What are unary and binary operators? To perform arithmetic calculations or operations arithmetic operators are used. Addition (+), subtraction (-), multiplication (*), division (/) returns quotient after division, modules (%) returns remainder after division.

Unary operator→ need one operand to act on like +a or –b; **Binary operator→** needs two operands to act on like a+b;

32. Explain the behaviour of + arithmetic operator with strings and characters? The + arithmetic operator is used to add two numeric values arithmetically. Plus operator can also be used with strings and characters to join them. Exp: char first='R', second = 'A'; String join=first + second; (value of will be join → "RA") String res = "Lucknow" + "Cirty"; (value of res will be → "LucknowCity")

33. What are increment and decrement operators? Increment operator ++ if used with a variable will increase the value of the variable by one (a++ or ++a) Decrement operator - - if used with a variable will decrease the value of the variable by one (a- - or - -a) a++ or ++a is equivalent to a=a+1 and a- - or - -a is equivalent to a=a – 1.

34. Give difference between postfix and prefix increment and decrement operators? Increment and decrement operators come in two varieties. Prefix → where the increment or the decrement operator (++ or - -) is placed before the operand like: ++a or - - a; In prefix the value of the variable is first increased or decreased and then it is used. Postfix→ where the increment or the decrement operator (++ or - -) is placed after the operand like: a++ or a- -; In postfix the value of the variable is first used and then it is increased or decreased. They have higher preference of execution over arithmetic operators and are executed as they appear from left to right.

35. Is K++ similar to ++K? If these statements are used individually as K++ or ++K then both are similar, as in both the cases the value of the variable K will increase by one. But if these statements are used as a part of an expression then the result will be different. Example: Suppose K=10; K=K++; SOP(K); Value of K will be printed as 10, because first the value of K is transfer and then increased (Postfix). On the other hand K=10; K=++K; SOP(K); will print the value as 11, as first the value is increased and then stored (Prefix)

36. Give difference between X++ and X+1? In case of X++ increment operator is used thus the value of X will increase by one. In X+1 the value of X will not increase but 1 will be added to the value of X.

37. What are relational operators? Relational operators determine the relation among different operands. They return a Boolean type value (true/false or 0/1) after evaluation. < less than, <=

less than equal to, = = equal to, > greater than, >=greater than equal to and != not equal to. The relation operators have lower precedence than arithmetic operators.

38. What value is returned by relational expression? Boolean type value (true/false or 0/1)

39. Give difference between = and = = operator? = is an assignment operator. It is used to assign some value to a variable. = = is a relational operator. It is used to compare two values whether they are equal (same) or not, and return a boolean type value.

40. What rules are followed while executing an expression? The execution of an expression is done from left to right, executing the brackets first. Any increment (++) or decrement (--) is done first. Then Exponent is done and then *,%,/ whatever comes first and then + and – is done. Thus arithmetic expression follows BEDMAS rule where division (/) multiplication (*) and modulus(%) has the same execution hierarchy and is executed left to right whatever comes first. Thereafter→ less than, greater than, less than equal to, greater than equal to, equal to (= =), not equal to, && (and) logical operator, || (OR) logical operator, conditional / ternary operator (? :) and at last assignment is done.

40a. Evaluate 14%2*5? Left to right 14%2= 0 then 0*5 = 0.

40b. Evaluate 5*14%2? Left to right 5*14= 70 then 70%2 = 0

40c. Evaluate 15/2*5? Left to right 15/2= 7 then 7*5 = 35.

40d. Evaluate 5*15/2? Left to right 5*15= 75 then 75/2 = 37.

40f. Evaluate 5+6/2? Division first 6/2 = 3 then 5+3=8.

40g. Evaluate 5+9/2.0*2? Division first 9/2.0 = 4.5 then 4.5*2 = 9.0 then 5 + 9.0 = 14.0.

40h. Evaluate 5+9*3.0/2? Multiplication first 9*3.0=27.0 then 27.0/2=13.5 the 5+13.5 = 18.5

41. Explain different logical operators? &&(AND) operator combines two or more conditions and evaluates to true value only and only if all the conditions are true. || (OR) operator combines two or more conditions and evaluates to true value if any one of the condition is true. ! (NOT) operator works with only one condition and returns the opposite of the actual value returned by the condition. Example: 18>12 is true and !(18>12) is false.

42. Assignment Operator? = is used as an assignment operator, to assign one value to another. Example: int x, y, z; x = 9; y = 7; z = x+ y; z= z * 2; Assignment operators has the lowest preference of execution.

43. Explain Short hand operators? Shorthand operators are used to simplify the coding of certain type of assignment statements. Example: a = a+10; can be written as a+=10; thus **var = var** *operator* **expression** is same as **var** *operator* = **expression.** Shorthand removes repeated operand.

44. What does expression R+=10 means? R+=10 is a shorthand of / equivalent to R = R + 10. This means that the value of R is increased by 10.

45. Explain conditional/ternary operator with the help of an example? Conditional / ternary operator works like the if () statement. Syntax: ValueVariable = Condition? expression1 : expression2; If the condition is true ValueVariable will hold the value returned by expression1 otherwise will hold the value returned by expression2. Example: int VV = 18>12? 9 : 3; The value of VV will be 9 because the condition is true. Ternary operator has lower presidency of execution than arithmetic operators.

46. Evaluate (14+ 11/2)%4? First the Parenthesis will be evaluated. Within brackets first 11/2 will be evaluated which returns 5. It is than added to 14 evaluating to 19. At last 19%4 will be evaluated → Ans. 3

47. What will be the value of x after evaluating: x= 10 + 12 * 15%5 – 10/2; Solution: Left to right (i) Multipy→ 10 + 180%5 – 10/2 (ii) Modules→ 10 + 0 – 10/2 (iii) Divide (/)→ 10 +

0 – 5 (iv) Add→ 10-5 (v) Subtract→ 5 thus answer is→ 5 **[**Find: a++ + 12 *15%++b - ++a/++c; if a=10,b=5,c=2**]**

48. Give the use of dot (.) operator? Dot operator (**.**) is used to access the data members and member functions of a class, with the help of associate object of that class.

49. Give the use of '*new*' operator? The *new* operator is used to reserve memory space for an instance variable, object and an array. Example: int arr[ ] = new int [10]; creates a new array. Give difference between expression and operator? Give example.

50. Explain some math functions available in java? What type of value do they return? Name the class under which they come.

Math.sin(x); Math.cos(x), tan(x), asin(y), acos(y), atan(y), atan2(x,y), pow(x,y), exp(x), log(x), sqrt(x), ceil(x), floor(x), rint(x), round(n), abs(a), max(a,b), min(a,b), Math.random() to generate random numbers. (x and y are double parameters, a and b may be int, long, float or double.) Mostly all return float / double type value or according to the passed parameter. **Math** is the class in java.lang package under which they come.

51. What is a pure integer expression, pure real expression and mixed expressions? In ***pure integer expression*** all operands are of integer type. Like→ int a,b,c; c = a + b; In ***pure real expression*** all operands are of floating or double type. Like→ float mks1, mks2, mks3, avg; avg = (mks1+mks2+mks3)/3; In ***mixed expression operands*** are of mixed or different data types. It returns the type after converting all operands upto the type of the largest operand. Exp. int a = 10; float b=12.5, double c = 15.25; then expression → (c + b * a) will return a double type value which is the largest type. From largest to smaller data types are : double → float→ int→char.

52. What does Type Conversion mean? The process of converting one predefined type into another is called type conversion.

53. What is implicit type conversion? The conversion of one pre-defined type into another by the compiler without programmer's intervention is known as ***implicit type conversion.*** It is applied to impure or mixed expressions.

54. What is Type Promotion? The compiler while evaluating an expression converts all operands upto the type of the largest operand, called ***type promotion.*** Exp.: int and float →float ; float and double → double ; char and int → int.

55. Define Coercion? The implicit type conversion where data types are promoted to their largest type is known as ***Coercion.*** Exp: if →int k=12; float t = 3; char ch='a'; then type returned by (k + t + ch) is float.

56. What is explicit type conversion or Type Casting? Explicit type conversion is user-defined that forces an expression to return a specified type, also known as type casting. Exp:. Math.pow(x,y); function always return a double type value. To force the function to return a integer type value we go for explicit type conversion→ int res = ***(int)*** Math.pow(x,y);

57. Give the importance of implicit type conversion or type promotion? Implicit type conversion is applied to mixed / impure expression and converts all the operands to upto the type of the largest operand so that there is no loss of information.

58. What is Boolean Expressions? Any expression that results into / returns a *false* or *true* value are called boolean expressions. Boolean expressions are combination of constants, variables and logical and relational operators. Example: (x>y); (y+z) >= (x/z); (y>x)||(z<y); (- y); are all boolean expressions.

59. What is a *statement* in java? Statements in java are complete unit of execution terminated with a semicolon(;). These are known as expression statements. ***Examples of different expression statements:***
int val = 12; //this is an declaration statement val++; //this is an increment expression
int num; // this is a variable declaration statement num=18; //this assignment statement
System.out.println(val); //this is a method call statement
ClassName obj = new ClassName(); or int arr[ ] = new int [12]; // these are object creation statements
The for() loop, while() loop, if() and switch() statements are examples of ***control flow statements*** that regulate the order in which the statements are executed.
60. What is a *Block*? A block is a group of zero or more executable statements between balanced braces. A block is represented or is formed by opening and closing curly braces. { }
60a. Compound Statements? Compound statements are a group of statements that are written with a block and are executed all at once when the block is executed.
61. What is a Null or Empty Statement? A null or empty statement is a statement that is ignored an is not executed by the compiler. It has no affect on the program coding or logic of the program. Any statement written after // signs is a null statement like→ // this is a empty or null statement
62. What does a class define? Give the significance of Classes? Class is a basic unit of objectoriented programming. It is described as a blue-print for an object. A class defines: The set of attributes, set of behaviour, how to construct and destroy. The ***class forms the basis of all computation*** because anything that exists as a part of java program has to exist as a part of a class, that is why Java is a **pure Object Oriented Language.** Here all functionalities revolve around classes and object, as in real world. The variables or objects, that are a part of a class, are operated upon by calling methods or functions.

5. Decision Making Statements

1. What are Decision Making Statements or selection control statements? These are *control flow statements* that regulate the order in which the statements are executed. Example ***if() else*** and ***switch() case*** statements. The execution of these statements depends upon a condition-test. These statements transfer the control and execute a block depending on the condition. They help in making decision. The flow of control is always from top to bottom in these statements. **Control Statements** are statements that controls the flow of statements. Like if() else, switch() case, ternary operator, for(), while() and do{}while();
2. Explain selection control statements with example? (i) *The if() else* statement checks a condition and depending on the *true/false* value returned by the condition, executes a certain block of statements.
Syntax: if(condition) { group of statements, executed if condition is true }
if(condition) { statements executed if condition is true } else { statements executed if condition is false }
(ii) *switch() case* is a multiple-branch selection statement. It tests a value against a list of integer or character constants. When a match is found it executes a particular ***case*** associated with that constant. Syntax: switch(expression) { case constant1 : statements; break; case constant2 : statements; break; and so on….; default : statements; } **explanation:** statements

of **case contant1** are executed if the value of switch expression matches with constant1. Statements of **case contant2** are executed if the value of switch expression matches with constant2. If no match is found then the statements of **default** are executed. The control is transferred out of switch block after execution of a case, due to the break statement.

3. Explain the concept of nested if () statement with example? A nested if() is an if() that has another if() in its if()s' body or in its else's body. Examples: if(condition1) { statements1… if(condition2) { statem2 …} else { statements2… } else { statements1… if(condition3) { statements3… } else { statements3… } }

4. What is dangling else? Dangling else arises when in a nested if statement, number of if() is more than the number of else clauses, then the last else clause is termed as dangling else. A dangling else statement goes with the immediately preceding unmatched if() statement. Example: if(ch>='A') if(ch<='Z') upper++; else others++; Here the else which is executed not when the outer if() is false but when the inner if() is false, is the dangling else clause.

5. How can we avoid default dangling else matching? The default dangling else matching can be overridden by using braces. Example: if(ch>='A') if(ch<='Z') upper++; else others++; (The idea is to execute the (dangling else) else clause if the outer if() is false. But in this case it will be executed when the inner if() is false. This can be overridden by keeping the inner if() within the braces of outer if(), like→ if(ch>='A') **{** if(ch<='Z') upper++; **}** else others++;

6. Explain Ladder if() statement with example? In a ladder-if type of if() structure the expressions are evaluated from the top to downwards. If at any point the expression becomes true the statement associated is with it is executed and the rest of the ladder is bypassed. If no condition is true than the last else is executed. If the last else is missing no action takes place. The name ladder-if or if-else staircase is given because its appearance is like a ladder.

Example: If(con1) if(con2) if(con3) exp3; else exp2; else exp1;

7. Which statement can be used as alternative of if() statement? Ternary/conditional operator [**? :**] is used as alternative of if() statement. Example: if(con1) exp1; else exp2; can be written as con1**?** exp1 **:** exp2;

8. Give differences between if() and ?: operator? The ternary operator is only good for short or small conditions, but if offers more clean and compact code. The if() is more flexible and nested if() can be used for multiple statements. Ternary can also be nested but it becomes complex and difficult to understand.

9. What is the use of *break* statement in switch–case? The break (jump statement), which is placed at the end of each case in switch-case transfers the control out of the switch block and does not lets you enter into any other case other than the matched case.

10. What is the role of default statement in switch-case? When it is executed? The default statement is the last and an optional statement of switch. It is executed when all the cases fail to match; if it is missing then no action takes place if all matches fail.

11. What is meant by *fall through* in switch case? In a switch-case if in the absence of a break statement the control flows to the next case below the matching case then it is known as fall through.

12. Can a *case* statement appear outside switch by itself? Why? No, A case statement cannot appear outside switch by itself, because case is a label which a part of switch.

13. Give the main differences between *switch-case* and *if-else* statements? If-else can handle ranges whereas switch cannot handle ranges. Switch can only handle integer and character test and cannot handle floating test. If() can evaluate a relational or logical expression, whereas switch can only test for equality against a set of constants only and not variables.

14. Explain nested-switch with a small example? A switch can be used within another switch within any case of outer switch. Switch within a switch is a nested witch. Example: *switch(a)* { case 1: *switch(b)* { case 1: statements; break; } break; …….. default: statements; }
15. In a nested-switch where will the control go after *breaking* from inner switch? Will it terminate all switch statements all only the current switch? In a nested switch the control will pass on to the outer switch block after breaking from the inner switch and will terminate only the current switch.
16. Is it always necessary to put a *break* after the *case* in a switch? When it can be avoided? It is very necessary to put a break after each case, though in the absence of break, it will not show any syntax error but it may lead to undesired result due to logical error. A break can be avoided for such type of problems where the same statement is to be executed for different matches. Example: switch(month){ case 2: days=28; break; case 4: case 6: case 9 : case 11 : days=30; break; default : days=31; }
17. Is it necessary to put a *break* after the *default* in a switch? Why? No, it is not necessary to put a break after the default statement, as it is the last statement of switch and the control will go out of the switch block automatically on its own.
18. What data types are supported by switch statement? Only integer type like: byte, short, int, long and character type char.
19. Can case labels in a switch have identical values? Two case labels in the same switch cannot have identical values. Only in a nested switch the case constants of the inner and the outer switch can contain common values.

6. Iteration through Loops

1. What is a looping / iteration statement? Give example? Iteration or loop is a construct that directs a program to execute a set of statements within a block, again and again as long as a specified condition is true. Once the condition becomes false the loop is terminated. Some of the looping statements are: for(); while() and do-while();
2. Explain the elements (parts of a loop) that control a loop? A loop has basically four parts. (i) Initialisation expression or expressions as they can be more than one. (ii) Test expression or condition that returns true or false and decides whether the loop will continue or stop. (iii) Update expression or expressions as they can be more than one. (iv) Block or the body of the loop enclosed within curly braces that contains zero or more statements that are to be executed again and again.
3. What is a control variable in a loop? A control variable in a loop is a variable that controls the loop. The execution of the loop depends on the initial value, condition and update of this variable. Example: for(k=1; k<=10; k++){ System.out.println(k); } Here, k is the control variable.
4. Which loop is best suited for fixed number of iterations? The for() loop is best suited for fixed number of iteration or when the number of iterations are known. The for() loop has the following syntax:
for(initialisation of variable; condition; updation) { block }
The given program will print numbers 1 to 10:→ for(int k=1; k<=10; k++){
System.out.println(k); }
5. Which loop is best suited if the number of iterations is unknown? The while() and do-while() loops are best suited when the number of iterations are unknown. Syntax of while and do-while() loops: ***while()loop:*** initialisation; while(condition) { updation } ***do-while()loop:***

initialisation; do {updation} while(condition);
Program to count the number of digits of a number inputted by the user: input n;
while(n>0){n=n/10; c++;}

6. Explain entry-control and exit-control loops? Give difference? In entry control loops (like for() and while()) the condition is checked first during the time of entering the block and allows execution of block only if the condition is true. But in a exit control loop (like dowhile()) the block is executed first without checking any condition because the test condition is placed at the end at the time of exit. Thus in this type of loop the block will be executed at least once, even if the condition is false.

7. Explain multiple initialisation expressions and multiple update expressions in a loop? A for() loop may contain ***multiple initialisation expressions and multiple update expressions***. These are separated by comma and are executed in a sequence. Example: for(k=1, sum=0; k<=number; sum=sum+k, k++) {System.out.print(k); } Here k=1 and sum=0 are multiple initialisation expressions and sum=sum+k and k++ are multiple update expressions.

8. What is meant by optional expressions in a for() loop? In a for() loop all the loop control expressions are optional i.e. the initialisation, condition and the updation. They can be omitted from for() statement and can be placed at other places in program coding. Initialisation can be done before the for(). Updation can be given within the block of for(). But semicolon will exist within the for(). Example: int k=1; for(; k<=10 ;) {
System.out.print(k); k++; } In this example the initialisation and the updation expression are skipped.

9. How can an infinite loop created? Give example? If we omit the test condition then an infinite for() loop is created. Example: for(k=1; ; k++) { System.out.print(k); } will run for infinite times. If all the three control expressions are omitted then an infinite loop is created. Exp.: for(;;) System.out.print("endless"); }

10. Explain empty loop with example? If a loop does not contains any statements within its block or loop body it is an empty loop. Example: for(k=1; k<=10; k++){ } or for(k=1; k<=10; k++);

11. Can a variable be declared within a loop? What will be its scope? A variable can be declared within a loop but its scope is limited to the block i.e. within the curly braces where it is declared and it cannot be accessed outside the block. Exp: for(k=1; k<=10; k++) { int sum = sum + k; } System.out.print(sum); Printing 'sum' outside the loop is invalid as its scope is only limited to the block where it is declared.

12. Give difference between while() and do-while() loop? While() is a entry controlled loop and do-while() is an exit control loop. *Ref. Question no. 6*

13. Explain the Jump statements: break, continue and return with example? The jump statements unconditionally transfer the program control from one point to another point in a program. The three types of jump statements are break, continue and return. **Return** is an unconditional jump statement. (1) Placed in a function the return causes an immediate exit from the function and the control is passed back to the function from where it was called. (2) It returns a value to the calling code. The **Break** makes the program to skip over part of the code. The break statement is used with switch(), for(), while() and do-while() loops and terminates the block when executed, and executes the statement following the block.
Example: While(test expression) { statement1; if(val>200) break; statement2; } statement3; Here statement3 will be executed as soon as the break statement written within if() is executed. The **Continue** statement also makes the program to skip over part of the code, but instead of

terminating the loop it transfers the control to the next iteration of the loop to take place, skipping any code in between. The continue statement can only be used with loops and not with switch(). Exp: for(int k=1; k<=10; k++) { statement1; if(k>5) continue; statement2; } statement3; In this example the program will not be able to execute statement2 when the value of k becomes more than 5.

14. Explain labels and goto jump statement with example? A label is a naming convention and lets you label statements. Labels are used on blocks and loops. Labelled blocks are useful with break and continue, and we can use break and continue statements with the labels. **A labelled break statement** causes the flow of control to jump out of the label mentioned with the break terminating the loop or block. **A labelled continue statement** causes the flow of control to jump to the next iteration of the labelled loop. (Ref. Example 10.8.3 on page 276) **Goto statement** lets the control of the program to transfer to the mentioned label. **Note: You are not allowed to use label and goto statements in the program.**

15. Explain nested loop with example? A loop within a loop is a nested loop. The control variables of the nested loops cannot be same. In a nested loop the inner loop is terminated first before the outer loop. Example: for(out=1; out<=5; out++) **{** for(in=1; in<=10; in++) **{** System.out.print((out*in)+" ");**}** System.out.println(); **}** This will print first 10 multiples for the first 5 numbers.

16. Explain the behaviour of jump and continue statement with nested loops? The break statement when used in the inner loop of nested structure then it will terminate only the current loop and the control is transferred to the outer loop. The continue will force the next iteration of the current loop in which it is placed.

7. Functions

What is a function or method? A function is a program having sequence of executable statements, which performs a specific task. Functions are also known as methods, procedures, sub-programs or sub-routines. A function may or may not return a value. A function can be a part of a class or can be a part of any other class or function.

Give advantage of using functions in a program?

Handles complex problems easily. Problem is divided into small manageable tasks. Hides low-level (less important) details. Reuses portions of code again and again without retyping them.

What type of function does not returns a value? Function with type **void** does not returns a value.

How many values can a function return at a time? Only one value can be returned by a function at a time.

What are the points kept in mind if the function returns a value? Types of value returned by function? If the function returns a value then its return type should either int, float, double, char, String or boolean. The function should end with a return statement followed by a variable of the same type as returned by the function.

What is the role of *return* statement in the function? Return is an unconditional jump statement. (1) Placed in a function the return causes an immediate exit from the function and the control is passed back to the function from where it was called. (2) It returns a value to the calling code.

What is meant by arguments or parameters? Arguments or parameters are those variables that appear in function definition and are passed through function.

Can a function be without arguments? Give example. Yes a function can be without arguments i.e. without parameters where no variables are passed through function. Example: void DrawLine() { for(int I=1;I<=80;I++) System.out.print("-"); } In this function no arguments are passed and it will draw a line when ever called.

What are conventions that should be followed while method naming? While naming a function we should see that name is meaningful and self descriptive. Function having multiple words are joined and begin with uppercase letter. Function name generally begin with a verb. Examples: printReportCard(); getMarks(); readData(); findFile(); isPrime(); getFactorial(); countVowels();

What is meant by function prototype? Give its importance. A Function Prototype: (i) describes the function, (ii) tells the program about the type of value returned by the function (return type) and (iii) tells about the type and number of argument passed (function signature).

What is meant by function signature? The argument passed through function is known as function signature. int getSum(int a,int b); [a and b]

How is a function executed or called? Explain with an example. Function can be called by the following ways:

Calling function of the same class

If the function returns a value: valueVariable = fucntionName();

Function without return value: functionName();

It is to be noted that if the function of the same class is to be called then it can be called just by giving the function name. But if a function of another class is to be called then an object is required which is the instance of that class i.e. an object representing a class.

Calling fn. of another class

If the function returns a value: value variable=obj.fnName();

Function without return value: obj.FnName();

Where obj is instance of class created by → className = new obj className();

What type of function can be used with an expression?

A function that returns a value can be used with an expression while calling.

(K = getSum(a,b) + 50;)

Explain Actual and Formal parameters with the help of an example?

Formal Parameters: Parameters that appear in function definition.

Actual Parameters: Parameters that appear in function call also known as original variables.

```
//function definition
int getSum(int a, int b) { int c = a + b; return c; }
Formal parameters.
//main function
void main()
{ int n1=10, n2=20;
int sum = getSum(n1,n2); //function calling
} Actual parameters
```

What type of arguments can be passed through function?

Arguments that can be passed through function are:

Primitive data types: char, byte, short, int, long, float and **Reference data types:** objects and arrays.

Explain the two manners of invoking/calling a function?

Function calling is done by two ways:

Calling function by VALUE method: Passing variables/values through function: In call by value method values of actual parameters are copied into the formal parameters i.e. function creates its own copy of argument values and works on these values. Whatever changes take place are not reflected back to the original or actual variables or parameters because function does not have access to the original variables. This type of calling is useful if original values are not to be modified.

Calling function by REFERENCE method: passing objects through function: In call by reference method in place of passing value to the function, a reference of the original variable is passed which stores the address of memory location of the variable. Thus the formal parameters become reference to the actual parameters in the calling function. Thus in this method the function does not creates its own copy but refers to the original values by a different name i.e. reference. Thus the called function works with the original data and any change in the values gets reflected in the original variables.

Explain the tree types of functions in terms of value returned by function:

1. Computational functions calculate or compute some value and return the computed value or result. Example: Math.pow(x,y)

2. Manipulative function manipulate information and returns a success or failure code denoted by 0 or 1; true or false. Example: string1.compareTo(string2)

3. Procedural function performs an action without any return value. Example: System.out.println();

Types of Functions in terms of modifying the formal parameters? Diff. of Pure and Impure function?

A **pure function** talks objects and primitives as arguments and does not modify them. The return value of a pure function is either a primitive or a value of a new object created inside the function.

An **impure function** changes / modifies the state of received object.

Explain Function overloading? Give Example.

What is function overloading? We can define several functions by the same name in the same class which are differentiated by the type or the number of argument passed through function. Example of function overloading:

```
class overload
{ int area(int s);
int area(float s);
int area(int l, double b);
int area(float h, double b);
}
```

Can there be many functions by the same name in a class? If yes what is the name give to this concept? Yes there can be more than one function by the same name in a class. This concept is known as function overloading (polymorphism)

8. Constructors

What is a constructor? Give example.

A constructor is a member function having a same name as that of its class. Constructor have no return type, not even void. **Exp→** Class xyz { int a, b; xyz( ){ a=10, b=30; } void display ( ) { System.out.print(a+"\t"+b); } }

What is the use of constructor in a class? A constructor is used to initialise the objects / data members of that class with a legal initial value.
How is a constructor executed or called? A constructor is executed on creation of an object for that class.
Can a constructor return any value? Constructor cannot return value, as it has no return type, not even void.
Explain the types of constructors with example: (A) Parameterised and (B) Non-Parameterised.
A constructor that receives parameters is a parameterised constructor. The data members of that class are initialised by the passed parameters.
Class xyz { int a, b; xyz(int n1, int n2)){ a=n1, b=n2; } void display ( ) {
System.out.print(a+"\t"+b); } }
A constructor that receives no parameters is a non-parameterised constructor. The data members of that class are initialised by default values or by given values.
Class xyz { int a, b; xyz( ){ a=0, b=0; } void display ( ) { System.out.print(a+"\t"+b); } }
Why we should define a constructor under the public section of a class? A constructor should be defines under the public section of a class so that its objects can be created in any function.
What is a default constructor? The compiler supplies a default constructor if it is not explicitly defined. The default constructor does not passes any arguments and it initialises the data members by any dummy values.
Explain Constructor overloading with the help of an example? When we have more than one constructor having different types and number of parameters in a class then its is known as constructor overloading.

```
class overLoad
{ static int a, b; static double c;
overLoad() // 1. non-parameterised constructor
{ a=0; b=0; c=0.0; }
overLoad(int x) // 2. parameterised constructor
{ a=x; b=x; c=0.0; }
overLoad(int x, int y, double z) // 3. parameterised constructor
{ a=x; b=y; c=z; }
void display() { System.out.println(a+"\t"+b+"\t"+c); }
void main()
{ overLoad ob1=new overLoad(); // # 1 constructor is called
display(); overLoad ob2=new overLoad(3); // # 2 constructor is called
display(); overLoad ob3=new overLoad(3, 6, 9.5); // # 3 constructor is called
display();
} } // Get the output for the above program code
```

What is the use of the *this* keyword? Explain with an example. Sometimes it happens that names of data member and the parameters passed through the constructor are same, i.e. the name of formal parameter or local variable is same as data member name of class. In that case the compiler will not be able to know as to which object / variable to manipulate. Such type of problem can be removed using the *this* keyword with the data members of the class. Example:
PROBLEM

```
class Box
```

{ int width, height, depth;
Box (int width, int height, int depth) // parameterised constructor
{ // now problem is which variable is being initialised
width = width; height = height; depth = depth;
}
// **SOLUTION** the above constructor should be written using t***his*** keyword as:
Box (int width, int height, int depth) // parameterised constructor
{ // Data members are referred by *this* operator
this.width = width; this.height = height; this.depth = depth; }

9. Class as User Defined Type

Define a class which is a Composite user defined Data Type and User Defined Data Types?
A **composite user defined data type** class has data members that are logically related and have functions that represent the behaviour of these data members.
Example:
Class date { int d,m,y; date() { d=1; m=1; y=2003; } date(int dd, int mm, int yy) { d=dd; m=mm; y=yy; }
void display() { System.out.println(d+"/"+m+"/"+y); } }
A **User defined data type** class does not has any data members but have functions that represent the behaviour of the class. Example: Class dog { void bark() {
System.out.println("WOOF! WOOF! "); } }
Here the behaviour of class dog is represented through method bark.
Give difference between primitive data types and user-defined data types?
Primitive data types are built-in data types. The sizes of these data types are fixed are these data types are available in all parts of a Java program. User-defined data types are created by the user. The size of these data types is equal to the total size of data members declared within the class. The availability of these data types depends upon their scope.
Explain the syntax of creating an object of a class?
ClassName objectName = new ClassName (arguments to constructor);
Constructor calling
How does a member of public specified class different from private class? Functions and the data members of a public class can be accessible from outside the class. Functions and the data members of a private class cannot be accessible from outside the class.

10. Using Library Classes

Give Difference between *System.out.print()* and *System.out.println()*?
The above methods are used to display values on the screen. The method *System.out.print()* will not transfer the control to the next line after print and thus the next successive print will appear on the same line. The method *System.out.println()* will transfer the control to the next line after print and thus the next successive print will appear on the next or newline.
What does System.in, System.out and System.err refer to? System.in is an input stream managed by the System class that implements the standard input stream. It is connected to the keyboard. System.out is the output stream for normal results connected to the monitor. System.err is the ouput stream for error messages also connected to the monitor.

Define Exception Handling? Exception handling refers to some unexpected situation or error that is unexpected that is not in the programmers control. Such type of errors that occur during programme execution are called exceptions and the way to handle them is called exception handling.

Explain different types of errors?

(a) Syntax errors occur when rules of a programming language are misused i.e. grammatical rule of program is violated. Exp: int main (int a, int b){ int c: c=a+b, Return c; } This program will give syntax error. 1_{st} line and the 2_{nd} line in the function is terminated by colon (:) and comma(,) instead of semicolon (;), moreover in the 3_{rd} line return statement being a keyword should be written in lower case.

(b) Semantics error occur when statements are not meaningful. Exp: the statement "Sita plays Guitar" is correct but "Guitar plays Sita" is semantically incorrect. Thus a + b = c is semantically incorrect.

(c) Type Errors is given when a wrong data type is given. Exp: int a="72" will give you type mismatch error as "72" is a string constant and variable **a** is a integer type variable.

(d) Run-time errors (execution errors) occurs during the execution of a program. Exp: is a program is trying to open a file that does not exist then execution error will occur. Similarly if enough memory is not available or an expression is trying to divide a number by zero are runtime errors.

(d) Logical errors cause a program to produce incorrect or undesired output. Exp: If we are trying to print 1_{st} 10 numbers and we write: int n = 1; while(n>=10) { System.out.println(n); n=n+1; } This program will not be executed at all as there is a logical error → while(n>=10) and initially n is 1 that is not greater than 10 thus this loop will not be executed. The correct way of writing is → while(n<=10)

What is the use of the *throws* keyword? The keyword *throws* is used to inform that an error has occurred. To handle such error we use *Exception* keyword immediately after *throws* keyword. Exp: void accept( ) throws Exception { }

What are wrapper classes? Give example. Wrapper classes are part of Java's standard library and these covert primitive data types in an object. A wrapper class wraps a value of primitive type in an object. Wrapper Class for **int** is **Integer**, for **char** is **Character** etc. We can see that primitive data types are all in lower case and wrapper class starts with a capital letter exp: **double** is a primitive type and **Double** is its wrapper class.

Give syntax to input following values in a variable

(i) integer value int num=Integer.parseInt(inv.readLine());

(ii) floating value float num=Float.parseFloat(inv.readLine());

(iii) double value double num= Double.parseDouble(inv.readLine());

(iv) character value char ch=(char)inv.read();

(v) String value String str=inv.readLine();

inv is an object created using either BufferedReader or DataInputStream

Give difference between the *String* objects and objects created by *String Buffer*?

Objects created using String Buffer are mutable (changeable) and a new object is not required.

Objects created using String are immutable (unchangeable) and a new object is required the hold changes. The original string remains unchanged.

Syntax to create a String Buffer Object: StringBuffer str = new StringBuffer(String variable);

String Buffer Functions: str.insert(position, "String"); str.reverse();

str.delete(stIndex, toIndex); str.setChar(position, char);

String Functions:
int len=str.length();
char ch=str.charAt(position);
String str=str1.concat(str2);
String str=str1+str2;
int p=str.indexOf(character/String);
int p=str.indexOf(character/String, n);
int p=str.lastIndexOf(character/String);
int k=str1.compareTo(str2);
String S=str.trim();
String S=str.substring(big, end);
String S=str.substring(big);
String S=str.toUpperCase();
String S=str.toLowerCase(); str=str.replace(char1,char2);
String S=str.valueOf(number); int
int num=Integer.parseInt("12345");
Boolean String Functions:
boolean res=str.startsWith(String);
boolean res=str.endsWith(String);
boolean res=str1.equals(str2);
boolean res=str1.equalsIgnoreCase(str2);
Character Functions:
char ch=Character.toLowerCase(ch);
char ch=Character.toUpperCase(ch);
boolean k = Character.isLetter(ch);
boolean k = Character.isDigit(ch);
boolean k = Character.isUpperCase(ch);
boolean k = Character.isLowerCase(ch);

Give difference between functions *indexOf()* and *lastIndexOf()*?
indexOf(char) → returns the index of the first occurrence of the given character from the string.
lastIndexOf(char) → returns the index of the last occurrence of the given character from the string.
indexOf(char,n) → returns the index of the first occurrence of the given character from n position.

Name some packages provided by Java? Packages are extensive library of pre-written classes that can be used in the programs. These classes are divided into groups called packages. Exp: java.applet; java.io (java input/output package); java.lang (java language package); java.util (java utility package)
Encapsulation (Chapter 11) Encapsulation is termed as Data Hiding, is a way of combining data and function (that operates on that data) under a single unit (class). Thus encapsulation hides the data from the outside world and does not allows direct access of data. Data can be only accessed by functions.
Visibility Modifiers: this defines which function or method is able to use this method.

Public : It can be called from outside the class also.
Private : Accessible within the class.
Protected : Methods in this class and the methods in any subclasses may use this method.
Scope and Visibility of a Variable/Object/class/function? The scope & visibility of a variable/object/ class/function is limited to the area within which it is declared. Area within opening and closing brasses i.e.{ }.
Exp: class xyz { int a; void main { int b; int accept() { int c; } } int display() { } }
Scope of variable **a** is limited within the { } of class xyz. It can be used in function main(), accept(), display().
Variable **b** declared within { } of main() can be used in function main(), accept() but not in display();
Variable **c** declared within { } of accept() can only be used within function accept() and not in any other fn.
The function accept() can only be used within main(), as it is declared within { } of main() function.
Global Variable: Class variable which is available to the entire class.
Local Variable: Variable declared inside a method or block { }
Variable **a** is known as global variable as it can be used within all function of the class area { }
Variable **b** is a local variable of function main(); Variable **c** is a local variable of function accept().

11. Arrays

Define array? Give syntax to declare array? An array is a collection of variables of the same type that are referenced by a common name. An array contains cells. Each of them is given a unique positive integer number beginning from 0,1,2,3…. and so on, which denotes the cell position known as index number. This number is enclosed within bracket (i.e. ***arr[0], arr[1], arr[2], arr[3],…..***) known as ***subscripted variable***. So we can say that array is a group of single subscripted variables. Each single subscripted variable can hold a value as stored in a normal variable. Single dimension array consists of a row or a column. Type ArrayName[]=new type [size]; Double dimension array consists of row and column. Type ArrayName[][]=new type [SizeRow][SizeCol]. Type can be int, float, double, char, String. Size can be any integer positive value.
How do we initialise single and double dimension arrays?
Initialise numbers 3,5,7,8,9 in an array? **int num[ ]={3,5,7,8,9};**
Initialise names "alok", "john", "sita", "ram" in an array? **String name[ ]={ "alok", "john", "sita", "ram"};**
Store 'k','d','t','m','e' in an array? **char ch[ ]={'k','d','t','m','e'};**
Store 4,5,3,2,1,6,7,8,9,5,6,2 in 3x4 2-D array? **int num[ ][ ]={** {4,5,3,2}, {1,6,7,8}, {9,5,6,2} **};**
Store "alok", "john", "sita", "ram", "rita","rash" in an 2x3 array? **String nam[ ][ ]={** {"alok", "john", "sita"}, {"ram", "rita","rash"} **};**
Values in an array can also be stored using a loops using input statements. We can also pass arrays through functions, to store the values. Arrays are by default passed by reference.
What is the need or advantage of using array in program?
When large volume of data of same type is to be stored for processing array is the best option. Values can be reused easily as all the data is referred by a single name. Program code is reduced and made easy to understand.

What is meant by the size of an array? What is the valid range of indices? The size of an array is equal to the number of cells referred by the single name. The valid range of indices is 0 to size-1. All element of an array are addressed using the name of the array and the subscripts 0,1,2,...n-1. If array is Z[5] then legal elements are Z[0], Z[1], Z[2], Z[3], Z[4]. Subscripts <0 or >=n are illegal and will be referred as out of bound subscripts.

What is the maximum cell address in an array having 10 cells? Ans: 9

Give difference between linear and binary searching? In linear searching the element to be searched is searched scanning each and every cell from 0. In binary searching the array elements should be arranged in some order. The value to be searched is first checked in the mid of array, if not then it is searched in the mid of that half to which it belongs and the process is repeated until value is found. This is faster then linear search.

Give difference between selection and bubble sorting? Sorting referrer to arranging the value in some order: ascending or descending order. If array contains 5 cells then, in selection sort the values are compared on one to one bases. [0] is compared with [1],[2],[3] and [4], then [1] is compared with [2],[3],[4] then [2] is compared with [3], and [4] and lastly [3] is compared with [4]. This process is repeated for 5 times in case of array having size 5. In bubble sort the values are compared in pairs. [in an array of size 5→ [0] is compared with [1], [1] with [2], [2] with [3] and [3] with [4]. This process is repeated 5 times. In bubble sort as the comparison is done in pairs thus the sorting becomes faster.

In a double dimension array the first subscript represents (row/col)? First subscript represents rows.

Give the advantages and disadvantages of using arrays? By the use of arrays large volume of similar type of values can be referred by a single name. It is easy to declare and use. The program code can be easily managed. The program code becomes small and easy to understand. Array elements can be randomly and directly accessed by providing the index number. The only disadvantage is that before using the array the size is to be known and we have to be careful that the array is declared to hold the largest possible group of data.

What are the points to be kept in mind while giving the size of an array?

While specifying the size of an array we have to be careful that the value should be a positive and integer. As the indexing of cells begins from zero so the last cell index is always one less than the size specified.

Programs for practice:

1. WAP to find the amount of discount and the final amount to be paid. When purchase amount is less than 5000 then discount rate is 10%. When purchase if between 5000 to 15000 discount rate is 20% and for any further purchases made the discount rate is 35%. Input purchases and display purchases made, discount allowed and the final payment.

2. A web site charges for sending emails by his customers according to the given conditions. For the first 200 e-mails send Rs. 50 is charged. Next 100 e-mails are charged @ Rs 1 for four e-mails. Any further e-mails are free of cost. Input number of e-mails send by a customer in a month and find the amount to be paid.

3. Calculate taxi charges after inputting distance in meters. For the first 1000 meters Rs. 15 is charged. Next 500 meters is charges @ Rs. 5 and every next 250 meters is charged @ Rs. 3

4. There are 40 students in a class. Find and print the frequency of students getting less than 40 marks, between 40 to <60 marks, greater than 60 but less than 75 marks and students getting 75 or above marks.

5. In an election, out of 1245 voters in a booth, only 963 voters use their franchise to vote. If five candidates are contesting, write a program to find:

i. number of valid and invalid votes.

ii. percentage of the valid votes received by each candidate.

6. WAP to accept net pay and category (A or B) and calculate tax based on the following conditions:

In category A → Net pay 10000 or below tax is 10% of net pay. Net pay above 10000 but below 25000 tax is 10.5% of net pay. Net pay 25000 or more tax is 11% of net pay.

In category B → Net pay 10000 or below tax is 10.25% of net pay. Net pay above 10000 but below 25000 tax is 12.5% of net pay and if net pay 25000 or more tax is 15% of net pay.

7. WAP to input a number and print appropriate message if it is prime or not.

8. WAP to print all the perfect numbers between 1 and the limited inputted by the user.

9. WAP to check if a number is an arm-strong number or not.

10. WAP to check if a number is automorphic or not.

11. WAP to print all three digit numbers *that are not ending with a zero*, in their reverse form along with original number. Also count total number of pallendrome numbers in the range.

12. WAP to print first 'n' terms of Fibonacci series.

13. Using switch case construct find and print the area of triangle, rectangle and product of two numbers depending on the users choice and other necessary inputs.

14. Using switch case store the numbers of days present in a month. Month number (1 to 12) is inputted by the user. (Try to use not more than 5 case labels and one default statement in the program)

15. WAP monthly telephone charges using the following information: Fixed monthly rent: Rs 380. Free calls during the month 120 calls. Charges per call beyond free limit upto 200 calls: Rs. 1.00. Charges per call in excess of 200+120 calls: Rs 1.25

16. Print sum of $1 + x + x!/2_2 + x!/4_3 + x!/6_4 + \ldots\ldots\ldots x!/n_t$

17. The production (P) of crude oil of a country in millions of barrels may be estimated by the following set of equations, where T represents the time in years. $P = 5 + 3T$, if $T <= 3$; $P = 14 +(T - 5/2)_2$, for $t > 3$. Write a program in JAVA to find the quantity of production for every year from T = 1 to 10 and print it.

18. WAP to find the factorial of an inputted number.

Functions:

Write a function, which returns Boolean type *true* if the passed number is prime otherwise returns *false*?

Write a function which prints a line of 'n' number of stars i.e. '*'.

Write a function to find the maximum of two numbers.

Program to show how to call a function of another class.

Complete the following functions:

int getFactorial(int) //fn. To return the factorial of a number

int countFactors(int) //fn. To return the count of factors of a number

int sumOfFactors(int) //fn. To return the sum of the factors of a number

int sumOfDigits(int) //fn. To return the sum of the digits of a number

int reverse(int) //fn. To return the reverse of a number

boolean isVowel(char) //fn. To check if a character is a vowel or not

boolean isPrime(int) //fn. To check if a number is a prime or not
boolean isPerfect(int) //fn. To check if a number is a perfect number or not
int isLeap(int) //fn. To check is a year is a leap year or not. Return 0 or 1
void spaces(int) //fn. That prints a given number of spaces
void ascendNum(int[ ]) //fn. To arrange array of numbers in ascending order
void alphaOrder(String[ ]) //fn. To arrange names in an array in alphabetic order
void meritList(name[], int mks[]) //fn arranges name in descending order of marks
int countSpaces(String) //fn. That returns the count of spaces of a string
Program to print twin prime nos. between 1 to 100 (use function isPrime(int))
Program to print all special nos. between 1 to 500 (use function getFactorial(int))
Program to check if the inputted number is magical no (use function sumOfDigits(int))
Program using function that returns an absolute value for any passed integer number.
Program to show that different objects of the same class store different values

Single Dimension Array:

1. Input N values in an array and find the sum and average of all the elements.
2. Arrange values in ascending and descending order using selection / bubble sort.
3. Search a given value using liner or binary search technique.
4. Search for maximum and minimum value along with its position.
5. Input names and print the name of maximum and minimum length.
6. (a) Input names and marks in two different arrays and print the name who got the highest and the lowest marks.
(b) Search a given name and print its marks.
(c) Print all the names getting marks above 70%.
(d) Arrange names according to marks in descending order.
(e) Arrange names in alphabetic order along with corresponding marks.
7. Quantity and rate for N products is stored in arrays and print a bill.

Double Dimension Array:

1. Print values in matrix form. Printing different given format.
2. Transposing of matrix.
3. Print matrix along with sum of rows and columns.
4. Print sum of main and secondary diagonal.
5. Sum of boundary elements, corner elements, elements above & below main diagonal.
6. Printing of elements above and below the main diagonals.
7. Search a value and print the row and column location.
8. Modify the cell according to row and column address.
9. Search and print the prime, Armstrong, automorphic etc nos.
10. Print the values that are at prime location.
11. Adding and subtracting the elements of 2-D array.

Series:

1. Fibonacci series and locus series.
2. Sum of numbers 1 to n
3. Product of numbers 1 to n (Factorial)
4. 1 + 3 + 5 + 7 + 9 + …… n
5. 2 + 4 + 6 + 8 + 10 + …..n
6. ½ + 1/3 + ¼ + 1/5 + 1/6 + …. 1/n
7. ½ + 2/3 + ¾ + 4/5 + 5/6 + ….n/n+1

8. 1 + x/2 + x/3 + x/4 + x/5 + ….x/n
9. 2! + 3! + 4! + 5! + 6! + …. N!
10. 1 + 3/2! + 4/3! + 5/4! + ….. n-1/n!
11. 2 – 3 + 4 – 5 + 6 – 7 + 8 – 9 …. N
12. (1+2) + (1+2+3) + (1+2+3+4) + (1+2+3+4+5)….. (1+2+3+4+5+……+n)
13. 1 + 3 – 5 + 7 – 9 + 11 – 13 + …… n

Patterns:

1. 1 12 123 1234 12345
2. 1 22 333 4444 55555
3. 12345 1234 123 12 1
4. 54321 4321 321 21 1
5. 5 45 345 2345 12345
6. 5 54 543 5432 54321
7. 1 21 321 4321 54321
8.

```
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * *
* * *
* *
*
```

9.

```
1
121
12321
1234321
123454321
```

10.

```
0
101
21012
3210123
432101234
```