* DATA STRUCTURE APTITUDE *

1. What is data structure?

A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other. Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

2. List out the areas in which data structures are applied extensively?

- Compiler Design,
- Operating System,
- Database Management System,
- Statistical analysis package,
- Numerical Analysis,
- Graphics,
- Artificial Intelligence,
- Simulation

3. What are the major data structures used in the following areas : RDBMS, Network data model & Hierarchical data model.

- RDBMS – Array (i.e. Array of structures)
- Network data model – Graph
- Hierarchical data model – Trees

4. If you are using C language to implement the heterogeneous linked list, what pointer type will you use?

The heterogeneous linked list contains different data types in its nodes and we need a link, pointer to connect them. It is not possible to use ordinary pointers

for this. So we go for void pointer. Void pointer is capable of storing pointer to any type as it is a generic pointer type.

5. Minimum number of queues needed to implement the priority queue?

Two. One queue is used for actual storing of data and another for storing priorities.

6. What is the data structures used to perform recursion?

Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls.

Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

7. What are the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?

Polish and Reverse Polish notations.

*8. Convert the expression ((A + B) * C – (D – E) ^ (F + G)) to equivalent Prefix and Postfix notations.*

Prefix Notation:

^ - * +ABC - DE + FG

Postfix Notation:

AB + C * DE - - FG + ^

9. Sorting is not possible by using which of the following methods?

(a) Insertion
(b) Selection
(c) Exchange
(d) Deletion

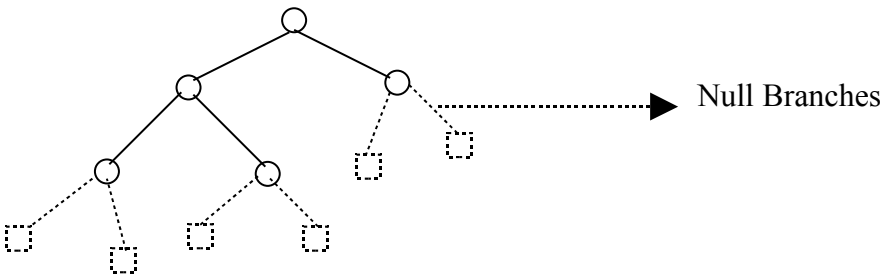
(d) Deletion.

Using insertion we can perform insertion sort, using selection we can perform selection sort, using exchange we can perform the bubble sort (and other similar sorting methods). But no sorting method can be done just using deletion.

10. A binary tree with 20 nodes has null branches?

21

Let us take a tree with 5 nodes (n=5)



It will have only 6 (ie,5+1) null branches. In general,

A binary tree with n nodes has exactly n+1 null nodes.

11. What are the methods available in storing sequential files?

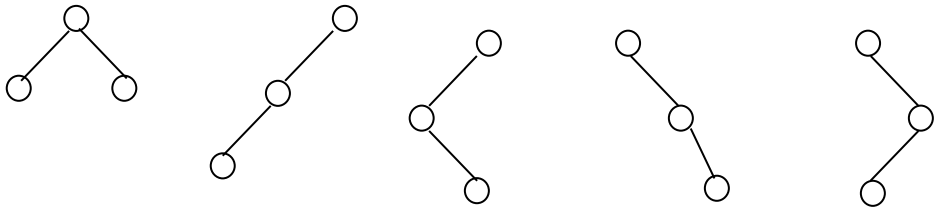
- Straight merging,
- Natural merging,

- Polyphase sort,
- Distribution of Initial runs.

12. How many different trees are possible with 10 nodes ?

1014

For example, consider a tree with 3 nodes(n=3), it will have the maximum combination of 5 different (ie, $2^3 - 3 = 5$) trees.



i ii

iii iv v

In general:

If there are n nodes, there exist 2^n-n different trees.

13. List out few of the Application of tree data-structure?

- The manipulation of Arithmetic expression,
- Symbol Table construction,
- Syntax analysis.

14. List out few of the applications that make use of Multilinked Structures?

- Sparse matrix,
- Index generation.

15. In tree construction which is the suitable efficient data structure?

(a) Array (b) Linked list (c) Stack (d) Queue (e) none

(b) Linked list

16. What is the type of the algorithm used in solving the 8 Queens problem?

Backtracking

17. In an AVL tree, at what condition the balancing is to be done?

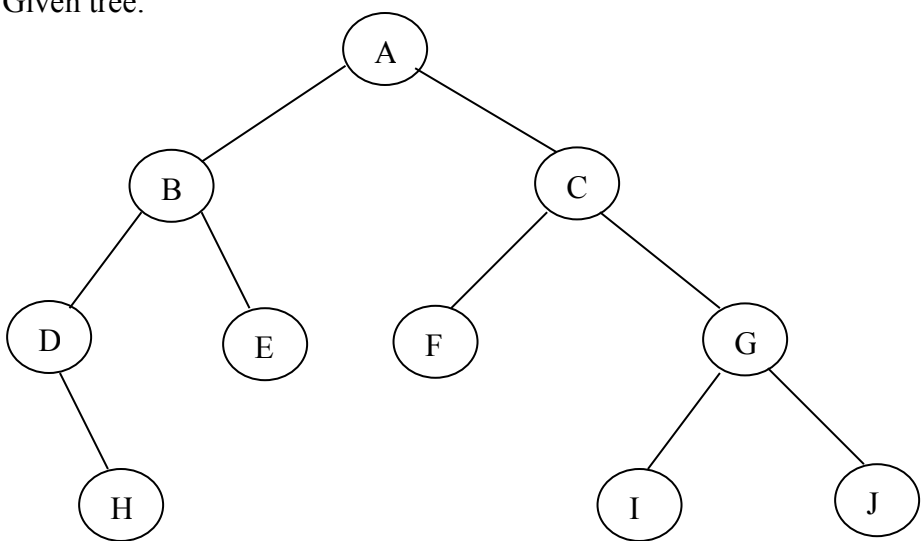
If the 'pivotal value' (or the 'Height factor') is greater than 1 or less than –1.

18. What is the bucket size, when the overlapping and collision occur at same time?

One. If there is only one entry possible in the bucket, when the collision occurs, there is no way to accommodate the colliding value. This results in the overlapping of values.

19. Traverse the given tree using Inorder, Preorder and Postorder traversals.

Given tree:



- Inorder : D H B E A F C I G J
- Preorder: A B D H E C F G I J
- Postorder: H D E B F I J G C A

20. There are 8, 15, 13, 14 nodes were there in 4 different trees. Which of them could have formed a full binary tree?

15.

In general:

There are 2^n-1 nodes in a full binary tree.

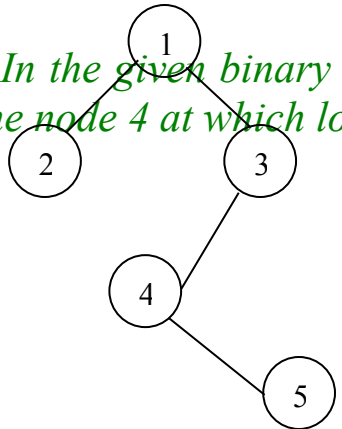
By the method of elimination:

Full binary trees contain *odd* number of nodes. So there cannot be full binary trees with 8 or 14 nodes, so rejected. With 13 nodes you can form a *complete* binary tree but not a full binary tree. So the correct answer is 15.

Note:

Full and Complete binary trees are different. *All full binary trees are complete binary trees but not vice versa.*

21. In the given binary tree, using array you can store the node 4 at which location?



At location 6

* RDBMS CONCEPTS *

1. *What is database?*

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2. *What is DBMS?*

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of *defining*, *constructing* and *manipulating* the database for various applications.

3. *What is a Database system?*

The database and DBMS software together is called as Database system.

4. *Advantages of DBMS?*

- Redundancy is controlled.

- Unauthorised access is restricted.
- Providing multiple user interfaces.
- Enforcing integrity constraints.
- Providing backup and recovery.

5. Disadvantage in File Processing System?

- Data redundancy & inconsistency.
- Difficult in accessing data.
- Data isolation.
- Data integrity.
- Concurrent access is not possible.
- Security Problems.

6. Describe the three levels of data abstraction?

The are three levels of abstraction:

- *Physical level:* The lowest level of abstraction describes how data are stored.
- *Logical level:* The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
- *View level:* The highest level of abstraction describes only part of entire database.

7. Define the "integrity rules"

There are two Integrity rules.

- *Entity Integrity:* States that “Primary key cannot have NULL value”
- *Referential Integrity:* States that “Foreign Key can be either a NULL value or should be Primary Key value of other relation.

8. What is extension and intension?

Extension -

It is the number of tuples present in a table at any instance. This is time dependent.

Intension -

It is a constant value that gives the name, structure of table and the constraints laid on it.

9. What is System R? What are its two major subsystems?

System R was designed and developed over a period of 1974-79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system.

Its two subsystems are

- Research Storage
- System Relational Data System.

10. How is the data structure of System R different from the relational structure?

Unlike Relational systems in System R

- Domains are not supported
- Enforcement of candidate key uniqueness is optional
- Enforcement of entity integrity is optional
- Referential integrity is not enforced

11. What is Data Independence?

Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify

the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

- Physical Data Independence: Modification in physical level should not affect the logical level.
- Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

12. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

14. What is E-R model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

15. What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

16. What is an Entity?

It is a 'thing' in the real world with an independent existence.

17. What is an Entity type?

It is a collection (set) of entities that have same attributes.

18. What is an Entity set?

It is a collection of all entities of particular entity type in the database.

19. What is an Extension of entity type?

The collections of entities of a particular entity type are grouped together into an entity set.

20. What is Weak Entity set?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

21. What is an attribute?

It is a particular property, which describes the entity.

22.

* RDBMS CONCEPTS *

1. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of *defining*, *constructing* and *manipulating* the database for various applications.

3. What is a Database system?

The database and DBMS software together is called as Database system.

4. Advantages of DBMS?

- Redundancy is controlled.
- Unauthorised access is restricted.
- Providing multiple user interfaces.
- Enforcing integrity constraints.
- Providing backup and recovery.

5. Disadvantage in File Processing System?

- Data redundancy & inconsistency.
- Difficult in accessing data.
- Data isolation.
- Data integrity.

- Concurrent access is not possible.
- Security Problems.

6. Describe the three levels of data abstraction?

The are three levels of abstraction:

- *Physical level:* The lowest level of abstraction describes how data are stored.
- *Logical level:* The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
- *View level:* The highest level of abstraction describes only part of entire database.

7. Define the "integrity rules"

There are two Integrity rules.

- *Entity Integrity:* States that "Primary key cannot have NULL value"
- *Referential Integrity:* States that "Foreign Key can be either a NULL value or should be Primary Key value of other relation.

8. What is extension and intension?

Extension -

It is the number of tuples present in a table at any instance. This is time dependent.

Intension -

It is a constant value that gives the name, structure of table and the constraints laid on it.

9. What is System R? What are its two major subsystems?

System R was designed and developed over a period of 1974-79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it

is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system.

Its two subsystems are

- Research Storage
- System Relational Data System.

10. How is the data structure of System R different from the relational structure?

Unlike Relational systems in System R

- Domains are not supported
- Enforcement of candidate key uniqueness is optional
- Enforcement of entity integrity is optional
- Referential integrity is not enforced

11. What is Data Independence?

Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

- Physical Data Independence: Modification in physical level should not affect the logical level.
- Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

12. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

14. What is E-R model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

15. What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

16. What is an Entity?

It is a 'thing' in the real world with an independent existence.

17. What is an Entity type?

It is a collection (set) of entities that have same attributes.

18. What is an Entity set?

It is a collection of all entities of particular entity type in the database.

19. What is an Extension of entity type?

The collections of entities of a particular entity type are grouped together into an entity set.

20. What is Weak Entity set?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

21. What is an attribute?

It is a particular property, which describes the entity.

22. What is a Relation Schema and a Relation?

A relation Schema denoted by R(A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ..., tn). Each tuple is an ordered list of n-values t=(v1,v2, ..., vn).

23. What is degree of a Relation?

It is the number of attribute of its relation schema.

24. What is Relationship?

It is an association among two or more entities.

25. What is Relationship set?

The collection (or set) of similar relationships.

26. What is Relationship type?

Relationship type defines a set of associations or a relationship set among a given set of entity types.

27. What is degree of Relationship type?

It is the number of entity type participating.

28. What is DDL (Data Definition Language)?

A data base schema is specifies by a set of definitions expressed by a special language called DDL.

29. What is VDL (View Definition Language)?

It specifies user views and their mappings to the conceptual schema.

30. What is SDL (Storage Definition Language)?

This language is to specify the internal schema. This language may specify the mapping between two schemas.

28. What is Data Storage - Definition Language?

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.

29. What is DML (Data Manipulation Language)?

This language that enable user to access or manipulate data as organized by appropriate data model.

- *Procedural DML or Low level:* DML requires a user to specify what data are needed and how to get those data.
- *Non-Procedural DML or High level:* DML requires a user to specify what data are needed without specifying how to get those data.

33. What is DML Compiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

34. What is Query evaluation engine?

It executes low-level instruction generated by compiler.

35. What is DDL Interpreter?

It interprets DDL statements and record them in tables containing metadata.

36. What is Record-at-a-time?

The Low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

37. What is Set-at-a-time or Set-oriented?

The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set-oriented.

38. What is Relational Algebra?

It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

39. What is Relational Calculus?

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL ALPHA, QUEL.

40. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus

The tuple-oriented calculus uses a tuple variables i.e., variable whose only permitted values are tuples of that relation. E.g. QUEL

The domain-oriented calculus has domain variables i.e., variables that range over the underlying domains instead of over relation. E.g. ILL, DEDUCE.

41. What is normalization?

It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

- Minimizing redundancy
- Minimizing insertion, deletion and update anomalies.

42. What is Functional Dependency?

A Functional dependency is denoted by X -> Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have

t1[Y] = t2[Y]. This means the value of X component of a tuple uniquely determines the value of component Y.

43. When is a functional dependency F said to be minimal?

- Every dependency in F has a single attribute for its right hand side.
- We cannot replace any dependency X -> A in F with a dependency Y-> A where Y is a proper subset of X and still have a set of dependency that is equivalent to F.
- We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

44. What is Multivalued dependency?

Multivalued dependency denoted by X -> Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties

- t3[x] = t4[X] = t1[X] = t2[X]
- t3[Y] = t1[Y] and t4[Y] = t2[Y]
- t3[Z] = t2[Z] and t4[Z] = t1[Z]
  where [Z = (R-(X U Y))]

45. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

46. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

47. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X-> Y is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

48. What is 2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

49. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X-> A either of the following is true

- X is a Super-key of R.
- A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

50. What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies additional constraints that for every FD X -> A, X must be a candidate key.

51. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency X -> Y that holds over R, one of following is true

- X is subset or equal to (or) XY = R.
- X is a super key.

52. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2, ..., Rn} that holds R, one the following is true

- Ri = R for some i.
- The join dependency is implied by the set of FD, over R in which the left side is key of R.

53. What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

54. What are partial, alternate,, artificial, compound and natural key?

Partial Key:

It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.

Alternate Key:

All Candidate Keys excluding the Primary Key are known as Alternate Keys.

Artificial Key:

If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.

Compound Key:

If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

Natural Key:

When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

56. What is indexing and what are the different kinds of indexing?

Indexing is a technique for determining how quickly specific data can be found.

Types:

- Binary search style indexing
- B-Tree indexing
- Inverted list indexing
- Memory resident table
- Table indexing

57. What is system catalog or catalog relation? How is better known as?

A RDBMS maintains a description of all the data that it contains, information about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.

58. What is meant by query optimization?

The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.

59. What is join dependency and inclusion dependency?

Join Dependency:

A Join dependency is generalization of Multivalued dependency.A JD {R1, R2, ..., Rn} is said to hold over a relation R if R1, R2,

R3, ..., Rn is a lossless-join decomposition of R . There is no set of sound and complete inference rules for JD.

Inclusion Dependency:

An Inclusion Dependency is a statement of the form that some columns of a relation are contained in other columns. A foreign key constraint is an example of inclusion dependency.

60. What is durability in DBMS?

Once the DBMS informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

61. What do you mean by atomicity and aggregation?

Atomicity:

Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. DBMS ensures this by undoing the actions of incomplete transactions.

Aggregation:

A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.

62. What is a Phantom Deadlock?

In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.

63. What is a checkpoint and when does it occur?

A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

64. What are the different phases of transaction?

Different phases are

- Analysis phase
- Redo Phase
- Undo phase

65. What do you mean by flat file database?

It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

66. What is "transparent DBMS"?

It is one, which keeps its Physical Structure hidden from user.

67. Brief theory of Network, Hierarchical schemas and their properties

Network schema uses a graph data structure to organize records example for such a database management system is CTCG while a hierarchical schema uses a tree data structure example for such a system is IMS.

68. What is a query?

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

69. What do you mean by Correlated subquery?

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a *correlated subquery*.

A correlated subquery can be easily identified if it contains any references to the parent subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.

E.g. Select * From CUST Where '10/03/1990' IN (Select ODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

70.What are the primitive operations common to all record management systems?

Addition, deletion and modification.

71. Name the buffer in which all the commands that are typed in are stored

'Edit' Buffer

72. What are the unary operations in Relational Algebra?

PROJECTION and SELECTION.

73. Are the resulting relations of PRODUCT and JOIN operation the same?

No.

PRODUCT: Concatenation of every row in one relation with every row in another.

JOIN: Concatenation of rows from one relation and related rows from another.

74. What is RDBMS KERNEL?

Two important pieces of RDBMS architecture are the kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database

You might think of an RDBMS as an operating system (or set of subsystems), designed specifically for controlling data access; its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its table-space structures

.

75. Name the sub-systems of a RDBMS

I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management

76. Which part of the RDBMS takes care of the data dictionary? How

Data dictionary is a set of tables and database objects that is stored in a special area of the database and maintained exclusively by the kernel.

77. What is the job of the information stored in data-dictionary?

The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.

78. Not only RDBMS takes care of locating data it also

determines an optimal access path to store or retrieve the data

79. How do you communicate with an RDBMS?

You communicate with an RDBMS using Structured Query Language (SQL)

80. Define SQL and state the differences between SQL and other conventional programming Languages

SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.

81. Name the three major set of files on disk that compose a database in Oracle

There are three major sets of files on disk that compose a database. All the files are binary. These are

- Database files
- Control files
- Redo logs

The most important of these are the database files where the actual data resides. The control files and the redo logs support the functioning of the architecture itself.

All three sets of files must be present, open, and available to Oracle for any data on the database to be useable. Without these files, you cannot access the database, and the database administrator might have to recover some or all of the database using a backup, if there is one.

82. What is an Oracle Instance?

The Oracle system processes, also known as Oracle background processes, provide functions for the user processes—functions that would otherwise be done by the user processes themselves

Oracle database-wide system memory is known as the SGA, the *system global area* or *shared global area*. The data and control structures in the SGA are shareable, and all the Oracle background processes and user processes can use them.

The combination of the SGA and the Oracle background processes is known as an *Oracle instance*

83. What are the four Oracle system processes that must always be up and running for the database to be useable

The four Oracle system processes that must always be up and running for the database to be useable include *DBWR* (Database Writer), *LGWR* (Log Writer), *SMON* (System Monitor), and *PMON* (Process Monitor).

84. What are database files, control files and log files. How many of these files should a database have at least? Why?

Database Files

The database files hold the actual data and are typically the largest in size. Depending on their

sizes, the tables (and other objects) for all the user accounts can go in one database file—but that's not an ideal situation because it does not make the database structure very flexible for controlling access to storage for different users, putting the database on different disk drives, or backing up and restoring just part of the database.

You must have at least one database file but usually, more than one files are used. In terms of accessing and using the data in the tables and other objects, the number (or location) of the files is immaterial.

The database files are fixed in size and never grow bigger than the size at which they were created

Control Files

The control files and redo logs support the rest of the architecture. Any database must have at least one control file, although you typically have more than one to guard against loss. The control file records the name of the database, the date and time it was created, the location of the database and redo logs, and the synchronization information to ensure that all three sets of files are always in step. Every time you add a new database or redo log file to the database, the information is recorded in the control files.

Redo Logs

Any database must have at least two redo logs. These are the journals for the database; the redo logs record all changes to the user objects or system objects. If any type of failure occurs, the changes recorded in the redo logs can be used to bring the database to a consistent state without losing any committed transactions. In the case of non-data loss failure, Oracle can apply the information in the redo logs automatically without intervention from the DBA.

The redo log files are fixed in size and never grow dynamically from the size at which they were created.

85. What is ROWID?

The ROWID is a unique database-wide physical address for every row on every table. Once assigned (when the row is first inserted into the database), it never changes until the row is deleted or the table is dropped.

The ROWID consists of the following three components, the combination of which uniquely identifies the physical storage location of the row.

- Oracle database file number, which contains the block with the rows
- Oracle block address, which contains the row
- The row within the block (because each block can hold many rows)

The ROWID is used internally in indexes as a quick means of retrieving rows with a particular key value. Application developers also use it in SQL statements as a quick way to access a row once they know the ROWID

86. What is Oracle Block? Can two Oracle Blocks have the same address?

Oracle "formats" the database files into a number of Oracle blocks when they are first created—making it easier for the RDBMS software to manage the files and easier to read data into the memory areas.

The block size should be a multiple of the operating system block size. Regardless of the block size, the entire block is not available for holding data;

Oracle takes up some space to manage the contents of the block. This block header has a minimum size, but it can grow.

These Oracle blocks are the smallest unit of storage. Increasing the Oracle block size can improve performance, but it should be done only when the database is first created.

Each Oracle block is numbered sequentially for each database file starting at 1. Two blocks can have the same block address if they are in different database files.

87. What is database Trigger?

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can e defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written in PL/SQL.

88. Name two utilities that Oracle provides, which are use for backup and recovery.

Along with the RDBMS software, Oracle provides two utilities that you can use to back up and restore the database. These utilities are *Export* and *Import*.

The *Export utility* dumps the definitions and data for the specified part of the database to an operating system binary file. The *Import utility* reads the file produced by an export, recreates the definitions of objects, and inserts the data

If Export and Import are used as a means of backing up and recovering the database, all the changes

made to the database cannot be recovered since the export was performed. The best you can do is recover the database to the time when the export was last performed.

89. What are stored-procedures? And what are the advantages of using them.

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result to the client. Stored procedures are used to reduce network traffic.

90. How are exceptions handled in PL/SQL? Give some of the internal exceptions' name

PL/SQL exception handling is a mechanism for dealing with run-time errors encountered during procedure execution. Use of this mechanism enables execution to continue if the error is not severe enough to cause procedure termination.

The exception handler must be defined within a subprogram specification. Errors cause the program to raise an exception with a transfer of control to the exception-handler block. After the exception handler executes, control returns to the block in which the handler was defined. If there are no more executable statements in the block, control returns to the caller.

User-Defined Exceptions

PL/SQL enables the user to define exception handlers in the declarations area of subprogram specifications. User accomplishes this by naming an exception as in the following example:

ot_failure EXCEPTION;

In this case, the exception name is ot_failure. Code associated with this handler is written in the EXCEPTION specification area as follows:

```
EXCEPTION
    when OT_FAILURE then
                out_status_code          :=
g_out_status_code;
      out_msg       := g_out_msg;
```

The following is an example of a subprogram exception:

```
EXCEPTION
    when NO_DATA_FOUND then
     g_out_status_code := 'FAIL';
     RAISE ot_failure;
```

Within this exception is the RAISE statement that transfers control back to the ot_failure exception handler. This technique of raising the exception is used to invoke all user-defined exceptions.

System-Defined Exceptions

Exceptions internal to PL/SQL are raised automatically upon error. NO_DATA_FOUND is a system-defined exception. Table below gives a complete list of internal exceptions.

PL/SQL internal exceptions.

Exception Name	*Oracle Error*
CURSOR_ALREADY_OPEN	ORA-06511
DUP_VAL_ON_INDEX	ORA-00001
INVALID_CURSOR	ORA-01001
INVALID_NUMBER	ORA-01722
LOGIN_DENIED	ORA-01017
NO_DATA_FOUND	ORA-01403
NOT_LOGGED_ON	ORA-01012
PROGRAM_ERROR	ORA-06501

STORAGE_ERROR	ORA-06500
TIMEOUT_ON_RESOURCE	ORA-00051
TOO_MANY_ROWS	ORA-01422
TRANSACTION_BACKED_OUT	ORA-00061
VALUE_ERROR	ORA-06502
ZERO_DIVIDE	ORA-01476

In addition to this list of exceptions, there is a catch-all exception named *OTHERS* that traps all errors for which specific error handling has not been established.

91. Does PL/SQL support "overloading"? Explain

The concept of *overloading* in PL/SQL relates to the idea that you can define procedures and functions with the same name. PL/SQL does not look only at the referenced name, however, to resolve a procedure or function call. The count and data types of formal parameters are also considered.

PL/SQL also attempts to resolve any procedure or function calls in locally defined packages before looking at globally defined packages or internal functions. To further ensure calling the proper procedure, you can use the dot notation. Prefacing a procedure or function name with the package name fully qualifies any procedure or function reference.

92. Tables derived from the ERD

a) Are totally unnormalised
b) Are always in 1NF
c) Can be further denormalised
d) May have multi-valued attributes

(b) Are always in 1NF

93. Spurious tuples may occur due to

i. Bad normalization
ii. Theta joins
iii. Updating tables from join

a) i & ii b) ii & iii
c) i & iii d) ii & iii

(a) i & iii because theta joins are joins made on keys that are not primary keys.

94. A B C is a set of attributes. The functional dependency is as follows

AB -> B
AC -> C
C -> B

a) is in 1NF
b) is in 2NF
c) is in 3NF
d) is in BCNF

(a) is in 1NF since $(AC)^+ = \{ A, B, C\}$ hence AC is the primary key. Since C B is a FD given, where neither C is a Key nor B is a prime attribute, this it is not in 3NF. Further B is not functionally dependent on key AC thus it is not in 2NF. Thus the given FDs is in 1NF.

95. In mapping of ERD to DFD

a) entities in ERD should correspond to an existing entity/store in DFD

b) entity in DFD is converted to attributes of an entity in ERD

c) relations in ERD has 1 to 1 correspondence to processes in DFD

d) relationships in ERD has 1 to 1 correspondence to flows in DFD

(a) entities in ERD should correspond to an existing entity/store in DFD

96. A dominant entity is the entity

a) on the N side in a 1 : N relationship
b) on the 1 side in a 1 : N relationship
c) on either side in a 1 : 1 relationship
d) nothing to do with 1 : 1 or 1 : N relationship

(b) on the 1 side in a 1 : N relationship

97. Select 'NORTH', CUSTOMER From CUST_DTLS Where REGION = 'N' Order By CUSTOMER Union Select 'EAST', CUSTOMER From CUST_DTLS Where REGION = 'E' Order By CUSTOMER

The above is

a) Not an error
b) Error - the string in single quotes 'NORTH' and 'SOUTH'
c) Error - the string should be in double quotes
d) Error - ORDER BY clause

(d) Error - the ORDER BY clause. Since ORDER BY clause cannot be used in UNIONS

98. What is Storage Manager?

It is a program module that provides the interface between the low-level data stored in database, application programs and queries submitted to the system.

99. What is Buffer Manager?

It is a program module, which is responsible for fetching data from disk storage into main memory and deciding what data to be cache in memory.

100. What is Transaction Manager?

It is a program module, which ensures that database, remains in a consistent state despite system failures and concurrent transaction execution proceeds without conflicting.

101. What is File Manager?

It is a program module, which manages the allocation of space on disk storage and data structure used to represent information stored on a disk.

102. What is Authorization and Integrity manager?

It is the program module, which tests for the satisfaction of integrity constraint and checks the authority of user to access data.

103. What are stand-alone procedures?

Procedures that are not part of a package are known as stand-alone because they independently defined. A good example of a stand-alone procedure is one written in a SQL*Forms application. These types of procedures are not available for reference from other Oracle tools. Another limitation of stand-alone procedures is that they are compiled at run time, which slows execution.

104. What are cursors give different types of cursors.

PL/SQL uses cursors for all database information accesses statements. The language supports the use two types of cursors

- *Implicit*
- *Explicit*

105. What is cold backup and hot backup (in case of Oracle)?

- *Cold Backup:*

It is copying the three sets of files (database files, redo logs, and control file) when the instance is shut down. This is a straight file copy, usually from the disk directly to tape. You must shut down the instance to guarantee a consistent copy.

If a cold backup is performed, the only option available in the event of data file loss is restoring all the files from the latest backup. All work performed on the database since the last backup is lost.

- *Hot Backup:*

Some sites (such as worldwide airline reservations systems) cannot shut down the database while making a backup copy of the files. The cold backup is not an available option.

So different means of backing up database must be used — the hot backup. Issue a SQL command to indicate to Oracle, on a tablespace-by-tablespace basis, that the files of the tablespace are to backed up. The users can continue to make full use of the files, including making changes to the data. Once the user has indicated that he/she wants to back up the tablespace files, he/she can use the operating system to copy those files to the desired backup destination.

The database must be running in ARCHIVELOG mode for the hot backup option.

If a data loss failure does occur, the lost database files can be restored using the hot backup and the online and offline redo logs created since the

backup was done. The database is restored to the most consistent state without any loss of committed transactions.

106. What are Armstrong rules? How do we say that they are complete and/or sound?

The well-known inference rules for FDs

- Reflexive rule :

  If Y is subset or equal to X then X -> Y.

- Augmentation rule:

  If X -> Y then XZ -> YZ.

- Transitive rule:

  If {X->Y, Y -> Z} then X -> Z.

- Decomposition rule :

  If X -> YZ then X-> Y.

- Union or Additive rule:

  If {X -> Y, X -> Z} then X -> Y Z.

- Pseudo Transitive rule :

  If {X -> Y, WY -> Z} then WX -> Z.

Of these the first three are known as Amstrong Rules. They are sound because it is enough if a set of FDs satisfy these three. They are called complete because using these three rules we can generate the rest all inference rules.

107. How can you find the minimal key of relational schema?

Minimal key is one which can identify each tuple of the given relation schema uniquely. For finding the minimal key it is required to find the closure that is the set of all attributes that are dependent on any given set

of attributes under the given set of functional dependency.

Algo. I Determining X^+, closure for X, given set of FDs F

1. Set $X^+ = X$
2. Set Old $X^+ = X^+$
3. For each FD Y -> Z in F and if Y belongs to X^+ then add Z to X^+
4. Repeat steps 2 and 3 until Old $X^+ = X^+$

Algo.II Determining minimal K for relation schema R, given set of FDs F

1. Set K to R that is make K a set of all attributes in R
2. For each attribute A in K
   a. Compute $(K - A)^+$ with respect to F
   b. If $(K - A)^+ = R$ then set $K = (K - A)^+$

108. What do you understand by dependency preservation?

Given a relation R and a set of FDs F, dependency preservation states that the closure of the union of the projection of F on each decomposed relation Ri is equal to the closure of F. i.e.,

$$((\Pi_{R1}(F)) \cup \ldots \cup (\Pi_{Rn}(F)))^+ = F^+$$

if decomposition is not dependency preserving, then some dependency is lost in the decomposition.

109. What is meant by Proactive, Retroactive and Simultaneous Update.

Proactive Update:

The updates that are applied to database before it becomes effective in real world.

Retroactive Update:

The updates that are applied to database after it becomes effective in real world.

Simultaneous Update:

The updates that are applied to database at the same time when it becomes effective in real world.

110. What are the different types of JOIN operations?

Equi Join: This is the most common type of join which involves only equality comparisons. The disadvantage in this type of join is that there.

---

more.....

```
1.#include <stdio.h>

void main()
{
int x = 5;

printf("%d,%d,%d",x,(x<<2),(x>>2));
}
Ans: 5,20,1

2.#include <stdio.h>

void main()
{
char a[]="hello";
```

```
char b[]="hai";
a=b;
printf("%s,%s",a,b);

}
```
ANs : since a is the base address of the array it is a lValue(constant)
which can not be assigned with a value a=b is a error.

```
3.#include <stdio.h>

void main()
{
char * p = "hello",ch;
ch = *++p;
printf("%c,%s",ch,p);

}
```

Ans e,ello

```
4.#include <stdio.h>
#include <string.h>
#define TOTAL 3
#define MAXNAMELEN 80
struct company
{
char organization[TOTAL][MAXNAMELEN];
};
int main()
{
void nameswap(struct company)
struct company x;
int i;
strcpy(x.organisation[0],"AATT India");
```

```
strcpy(x.organisation[1],"AATT Corporation");
strcpy(x.organisation[2],"AATT Limited");
nameswap(x);
for(i=0;i<TOTAL;i++)
printf("\n %s",x.organisation[i]);
return(1);
}
void nameswap(struct company x)
{
char tempname[MAXNAMELEN];
strcpy(tempname,x.organisation[0]);
strcpy(x.organisation[0],x.organisation[1]);
strcpy(x.organisation[1],tempname);
return;
}
```

Ans : AATT India
AATT Corporation
AATT Limited

```
5 #inlcude <stdio.h>
#define Stringizer(x) printf(#x)
void main()
{
Stringizer(hello);
}
```

Ans: '#' is called Stringizer Operator - #x converts x to String Constant

```
6. #include <stdio.h>
#define Charizing(x) printf("%d",#@x)
void main()
{
Charizing(a);
```

```
}
```

Ans : 97

7.
```
#include<stdio.h>
#include<string.h>
int main( )
{
char *ptr1,*ptr2;
ptr1 = "Hello AATT";
ptr2 = "Hai";
ptr1= strcat(ptr1,ptr2);
printf("\n Input string %s",ptr1);
return 1;
}
```

Ans : Enough memory not allocated for concatenation

8.
```
int main( )
{
int x=20,y=35;
x=y++ + x++;
y=++y + ++x;
printf("%d %d",x,y);
return 1;
}
```

Ans : 57,94

9.
```
int main( )
{
int i,*p;
i=10;
p=&i;
printf("%d",10/*p);
return 1;
}
```

Ans:Since there is no space between / and * it is

taken to be a comment and the output will not be 10/10 =1 as expected

10. #include <stdio.h>

```
int main()
{
int a=0;
if(a= 0)
printf("hello");
printf("AATT");
return 0;
}
```

Ans :AATT

11.Write a function to swap two numbers without using temp .

Ans :

```
Swap(a,b)
{
a= a+b;
b= a-b;
a= a-b;
}
```

12.O/p of Following

```
int main()
{
char a[2];
*a[0]=7;
*a[1]=5;
printf("%d",&a[1]-a);
}
```

Ans illegal indirection

13.

```
int main()
{
char a[]="hellow";
char *b="hellow";
char c[5]="hellow";
printf("%s %s %s ",a,b,c);
printf(" %d,%d,%d",sizeof(a),sizeof(b),sizeof(c));
}
```

ans : 7,4,5 (Size of b is dependent on the machine)

14.

```
#define min(a,b) ((a)<(b))?(a):(b)
main()
{
int i=0,a[20],*ptr;
ptr=a;
while(min(ptr++,&a[9])<&a[8])
i=i+1;
printf("i=%d\n",i);
}
```

Ans :5

15.Prog to find PrimeFactors

```
void Prime( int);

int main()
{
Prime(56);
}
```

```
void Prime (int a)
{
int j;
for(j=1; j<=a; j++)
{
if(a% j == 0)
{
a/=j;
printf("%d,",j);
j= 1;
}
}
return ;
}
```

16 Function to Reverse the string .

```
char * reverse(char a[])
{
int len ,i=0;
char * p;
len = strlen(a);

p =(char *) malloc(len+1);

while(i<= len)
{

p[i++] = a[len -i];

}
p[len] = '\0';
return p;
}
```

17. Output of

```
int main()
{
printf("%d%x\n",0x2,12);

}
```

ans :2C

18.swap two var without Temp and Arithmetic ops

Ans:

```
swap(a,b)
{
a=a^b;
b=a^b;
a=a^b;
}
```

19.

```
int main()
{
int a[5],*p;
for(p=a;p<&a[5];p++)
{
*p=p-a;
printf("%d\n",*p);
}

}
```

Ans : 0,1,2,3,4

20.Prg to Reverse A number

```
Reverse(n)
{
int result = 0;
```

```
while(n!=0)
{ temp = n % 10;
result =result * 10 + temp
n = n/10;
}
return result;
}
```

21. int zap(int n)

```
{
if(n<=1)then zap=1;
else zap=zap(n-3)+zap(n-1);
}
```

then the call zap(6) gives the values of zap
[a] 8 [b] 9 [c] 6 [d] 12 [e] 15
Ans b

22.op for following prg

```
int main()
{
char *x="new";
char *y="dictonary";

void swap (char * , char *);
swap (x,y);
printf("(%s, %s)",x,y);

}
void swap (char *x,char *y)
{
char *t;
t=x;
x=y;
y=t;
printf("(%s, %s)-",x,y);
```

```
}
```

a).(New,Dictionary)-(New,Dictionary)
b).(Dictionary,New)-(New,Dictionary)
c).(New,Dictionary)-(Dictionary,New)
d).(Dictionary,New)-(Dictionary,New)
e).None of the above
Ans :c
23.

```
main()
{
main();
printf("InMain \t");
}
```

Ans : No Statement will be printed on the screen.

24.

```
main()
{
printf("InMain \t");
main();
}
```

Ans ???

25.

```
void strcp1(char *s,char *t)
{
while(*t)
*s++ = *t++;
}
main()
{
char a[] = "God is great";
char b[] = "god is man or god";
strcp1(b,a);
```

```
printf("%s",b);
}
```

Ans :GOD IS GREAT man or god

26.

```
int x ;
int intg()
{
return x+=4;
}
int diff()
{
return x/=4;
}

main()
{
x = intg()+diff();
printf("%d",x);
}
```

Ans 5

27.

```
int x;
main()
{
int j =2;
printf("%d",fun(j));
}

fun(int x)
{
x++,++x;
return x++;
```

```
}
```

Ans 4

28.

```
main()
{
static char *s[] = {"ice","green","cone","please"};
static char **ptr[] = {s+3,s+2,s+1,s};
char ***p = ptr;
printf("%s",**++p);
printf("\n%s",*--*++p +3);
printf("\n%s",*p[-2] +3);
printf("\n%s" , p[-1][-1] +1);
}
```

Ans :

```
cone

ase
reen
```

29.

```
main()
{
float **a;
int i,j;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
scanf("%f",&a[i][j]);
for(i=0;i<3;i++)
for(j=0;j<3;j++)
printf("%f",a[i][j]);
}
```

Ans :
Memory not allocated for a

30.

```
main()
{
printf("%d",sizeof("1234"));
printf("%d",sizeof(1));
printf("%d",sizeof(1.1));
}
```

Ans 548

31.

```
main()
{
int *x;
int y[3] = {10,20,30};
x=y;
printf("%d",++*x);
printf("\n%d",++*x++);
}
```

Ans 11,12

32.

```
int main()
{
char s[] = "Get organised";
printf("%s",&s[2]);
printf("\n %s",s);
printf("\n %s",&s);
printf("\n %c",s[2]);
```

```
}
```

Ans:

```
t organised
Get organised
Get organised
t
```

33.

What is the diff between the
i) int const *k;
ii) int * const k;

Ans : i) specifies that k is a pointer to integer constant , the k can be changed to point to another int value,but the value pointed by k can not be changed.
ii) specifies that k is a constant pointer to an integer , the value pointed to by the k can be changed ,but the k can not be made to point to another integer .

34 .

```
main()
{
int const *f;
int x = 10;
f = & x;
*f = 15
printf("%d",x);

}
```

Ans :

f cannot change the value of the pointed memory.

35.

```
main()
{
int const *f;
int x = 10,y =15;
f = & x;
printf("%d",*f);
f = & y;
printf("%d",*f);

}
```

Ans :
f cannot be assigned different addresses because it is a constant pointer.

36.

```
main()
{
int a[6]={0,1,2,3,4,5};
int (*p)[3];
p=a;
*p[1] = 5;
printf("%d",a[1]);
}
```

37.

```
main()
{
char i ;
for(i=0;i<256;i++)
printf("%d",i);
```

```
}
```

Ans :Infinite loop

38.

```
main()
{
char s[]="Hello World";
s++;
printf("%s",s);
}
```

Ans :S is an l value ,cannot be a target of an assignment stmt.

39.

```
main()
{
char *s="Hello World";
s++;
printf("%s",s);
}
```

Ans : ello world

40.

```
main()
{
char *s;
s = "Hello World";
printf("%s",s);
}
```

Ans :Hello World

41.

```
main()
{
int *s;
s = "Hello World";
printf("%s",s);
}
```

Ans :Hello World

42.

```
main()
{
char s[] = "Hello";
printf("%s" , s[3]);
}
```

Ans : %s needs an address whereas s[3] returns a content in the location s+3.

43.

```
main()
{
extern int a;
printf("%d",a);
}
```

Ans : Linkage error .

44.

```
main()
{
inti=-3,j=2,k=0,m;
m=++i&&++j||++k;
printf("\n %d %d %d %d",i,j,k,m);
```

```
}
```

Ans : 2 3 0 1

45.

```
main()
{
int a,b;
a=sumdig(123);
b=sumdig(123);
printf("%d %d",a,b);
}
sumdig(int n)
{
static int s=0;
int d;
if(n!=0)
{
d=n%10;
n=(n-d)/10;
s=s+d;
sumdig(n);
}
else return(s);
}
```

Ans :6,12

46.

```
#define CUBE(x) (x*x*x)
main()
{
int a,b=3;
a=CUBE(b++);
printf("\n %d %d",a,b);
```

```
}
```

Ans :27,6

47.

```
main()
{
char *p,*f();
p=f();
printf("f() returns:%s\n",p);
}
char *f()
{
char result[80];
strcpy(result,"anything will do");
return (result);
}
```

Ans : f() returns :

48.

```
main()
{
short int *p,*q;
p=(short int *)1000;
q=(short int *)2000;
printf("%d",(q-p));
}
```

Ans 1000/2 = 500

49

```
main()
{
char a =0xAA ;
int b ;
```

```
b = (int) a ;
b = b >> 4 ;
printf("%x",b);

}
```

Ans : ffffffaa

50.

```
.main()
{
char *p = "hello world!";
p[0] = 'H';
printf("%s",p);
}
```

Ans : Run time error .

51.

```
main()
{
char *p ;
char s[20] = "Hello world!";
strcpy(p,s);
printf("%s",p);
}
```

Ans : Run time error

52.

```
main()
{
char *p ;
char s[20] = "Hello world!";
```

```
p = (char *)malloc(20);
strcpy(p,s);
printf("%s",p);
}
```

Ans : Hello World!

53.

```
main()
{
int a=10,b;
a>= 5 ? b=100 : b=200;
printf("\n%d",b);
}
```

Ans :lvalue required

54.

```
main()
{
int a=10,b;
a>5 ? a++ : a++;
printf("\n%d",a);
}
```

Ans : 11

55.
Given a piece of code

```
int x[10];
int *ab;
ab=x;
```

To access the 6th element of the array which of

the following is incorrect?
(A) *(x+5) (B) x[5] (C) ab[5] (D) *(*ab+5} .

Ans : d

56.
Consider the following program
main()
{
int a[5]={1,3,6,7,0};
int *b;
b=&a[2];
}
The value of b[-1] is
(A) 1 (B) 3 (C) -6 (D) none

Ans : b
57.
What does the following program print?
#include <stio.h>
int sum,count;
void main(void)
{
for(count=5;sum+=--count;)
printf("%d",sum);
}
a. The pgm goes to an infinite loop b. Prints
4791010974 c. Prints 4791001974
d. Prints 5802112085 e. Not sure

Ans: a

58.
#include <stdio.h>
void main(void)

```
{
int i;
for(i=2;i<=7;i++)
printf("%5d",fno());
}
fno()
{
staticintf1=1,f2=1,f3;
return(f3=f1+f2,f1=f2,f2=f3);
}
```

a. produce syntax errors b. 2 3 5 8 13 21 will be displayed c. 2 2 2 2 2 2 will be displayed
d. none of the above e. Not sure

Ans : b

59.

```
#include <stdio.h>
void main (void)
{
int x;
x = 0;
if (x=0)
printf ("Value of x is 0");
else
printf ("Value of x is not 0");
}
```

a. print value of x is 0 b. print value of x is not 0 c. does not print anything on the screen
d. there is a syntax error in the if statement e. Not sure

Ans : a

60.

```
void main (void)
{
char arr[100] = {"Welcome to Mistral"};
foo (arr);
}
foo (char *x)
{
printf ("%d\t",strlen (x));
printf ("%d\t",sizeof(x));
return0;
}
```

a. 100 100 b. 18 100 c. 18 18 d. 18 2 e. Not sure

Ans : 18,100

61.

```
#include <stdio.h>
display()
{
printf ("\n Hello World");
return 0;
}
void main (void)
{
int (* func_ptr) ();
func_ptr = display;
printf ("\n %u",func_ptr);
(* func_ptr) ();
}
```

Ans : it prints the address of the function display and prints Hello World on the screen

62.

```
#include <stdio.h>
void main (void)
{
int i = 0;
char ch = 'A';
do
putchar (ch);
while(i++ < 5 || ++ch <= 'F');
}
```

Ans : AAAAAABCDEF

---

* C++ PROGRAMS *

Note: All the programs are tested under Turbo C++ 3.0, 4.5 and Microsoft VC++ 6.0 compilers.

It is assumed that,

- Programs run under Windows environment,
- The underlying machine is an x86 based system,
- Program is compiled using Turbo C/C++ compiler.

The program output may depend on the information based on this assumptions (for example sizeof(int) == 2 may be assumed).

1)
```
class Sample
{
public:
    int *ptr;
    Sample(int i)
    {
    ptr = new int(i);
    }
    ~Sample()
    {
```

```
        delete ptr;
        }
        void PrintVal()
        {
        cout << "The value is " << *ptr;
        }
};
void SomeFunc(Sample x)
{
cout << "Say i am in someFunc " << endl;
}
int main()
{
Sample s1= 10;
SomeFunc(s1);
s1.PrintVal();
}
```

Answer:

Say i am in someFunc
Null pointer assignment(Run-time error)

Explanation:

As the object is passed by value to SomeFunc the destructor of the object is called when the control returns from the function. So when PrintVal is called it meets up with ptr that has been freed.The solution is to pass the Sample object by *reference* to SomeFunc:

```
void SomeFunc(Sample &x)
{
cout << "Say i am in someFunc " << endl;
}
```

because when we pass objects by refernece that object is not destroyed. while returning from the function.

2) Which is the parameter that is added to every non-static member function when it is called?

Answer:

'this' pointer

3)
```
class base
        {
        public:
        int bval;
        base(){ bval=0;}
        };

class deri:public base
        {
        public:
        int dval;
        deri(){ dval=1;}
        };
void SomeFunc(base *arr,int size)
{
for(int i=0; i<size; i++,arr++)
        cout<<arr->bval;
cout<<endl;
}

int main()
{
base BaseArr[5];
SomeFunc(BaseArr,5);
deri DeriArr[5];
SomeFunc(DeriArr,5);
}
```

Answer:

00000

01010

Explanation:

The function SomeFunc expects two arguments. The first one is a pointer to an array of base class objects and the second one is the sizeof the array. The first call of someFunc calls it with an array of base objects, so it works correctly and prints the bval of all the objects. When Somefunc is called the second time the argument passed is the pointer to an array of derived class objects and not the array of base class objects. But that is what the function expects to be sent. So the derived class pointer is promoted to base class pointer and the address is sent to the function. SomeFunc() knows nothing about this and just treats the pointer as an array of base class objects. So when arr++ is met, the size of base class object is taken into consideration and is incremented by sizeof(int) bytes for bval (the deri class objects have bval and dval as members and so is of size >= sizeof(int)+sizeof(int)).

4)
```
class base
    {
    public:
       void baseFun(){ cout<<"from base"<<endl;}
    };
 class deri:public base
    {
    public:
       void baseFun(){ cout<< "from derived"<<endl;}
    };
void SomeFunc(base *baseObj)
{
    baseObj->baseFun();
}
int main()
{
base baseObject;
SomeFunc(&baseObject);
```

```
deri deriObject;
SomeFunc(&deriObject);
}
```
Answer:

```
from base
from base
```
Explanation:

As we have seen in the previous case, SomeFunc expects a pointer to a base class. Since a pointer to a derived class object is passed, it treats the argument only as a base class pointer and the corresponding base function is called.

5)
```
class base
    {
    public:
      virtual void baseFun(){ cout<<"from base"<<endl;}
    };
 class deri:public base
    {
    public:
      void baseFun(){ cout<< "from derived"<<endl;}
    };
void SomeFunc(base *baseObj)
{
    baseObj->baseFun();
}
int main()
{
base baseObject;
SomeFunc(&baseObject);
deri deriObject;
SomeFunc(&deriObject);
}
```
Answer:

```
from base
```

from derived

Explanation:

Remember that baseFunc is a virtual function. That means that it supports run-time polymorphism. So the function corresponding to the derived class object is called.

6) void main()

```
  {
        int a, *pa, &ra;
        pa = &a;
        ra = a;
        cout <<"a="<<a <<"*pa="<<*pa <<"ra"<<ra ;
}
/*
```

Answer:

Compiler Error: 'ra',reference must be initialized

Explanation:

Pointers are different from references. One of the main differences is that the pointers can be both initialized and assigned, whereas references can only be initialized. So this code issues an error.

*/

7)

```
      const int size = 5;
      void print(int *ptr)
    {
      cout<<ptr[0];
    }

void print(int ptr[size])
{
      cout<<ptr[0];
}
```

```
void main()
{
	int a[size] = {1,2,3,4,5};
	int *b = new int(size);
	print(a);
	print(b);
}
/*
```
Answer:
	Compiler Error : function 'void print(int *)' already has a body

Explanation:
	Arrays cannot be passed to functions, only pointers (for arrays, base addresses)
can be passed. So the arguments int *ptr and int prt[size] have no difference
as function arguments. In other words, both the functions have the same signature and
so cannot be overloaded.
*/

8.
```
class some{
public:
	~some()
	{
		cout<<"some's destructor"<<endl;
	}
};

void main()
{
	some s;
	s.~some();
```

```
}
/*
Answer:
        some's destructor
        some's destructor
Explanation:
        Destructors can be called explicitly. Here 's.~some()'
explicitly calls the
destructor of 's'. When main() returns, destructor of s is called
again,
hence the result.
*/
```

9.

```
#include <iostream.h>

class fig2d
{
        int dim1;
        int dim2;

public:
        fig2d() { dim1=5; dim2=6;}

        virtual void operator<<(ostream & rhs);
};

void fig2d::operator<<(ostream &rhs)
{
        rhs <<this->dim1<<" "<<this->dim2<<" ";
}

/*class fig3d : public fig2d
{
```

```
        int dim3;
public:
        fig3d() { dim3=7;}
        virtual void operator<<(ostream &rhs);
};
void fig3d::operator<<(ostream &rhs)
{
        fig2d::operator <<(rhs);
        rhs<<this->dim3;
}
*/

void main()
{
        fig2d obj1;
//      fig3d obj2;

        obj1 << cout;
//      obj2 << cout;
}
/*
```

Answer :

```
        5 6
```

Explanation:

In this program, the << operator is overloaded with ostream as argument.
This enables the 'cout' to be present at the right-hand-side. Normally, 'cout'
is implemented as global function, but it doesn't mean that 'cout' is not possible
to be overloaded as member function.

Overloading << as virtual member function becomes handy when the class in which
it is overloaded is inherited, and this becomes available to be overrided. This is as opposed

to global friend functions, where friend's are not inherited.
*/

10.

```
class opOverload{
public:
	bool operator==(opOverload temp);
};

bool opOverload::operator==(opOverload temp){
	if(*this  == temp ){
		cout<<"The both are same objects\n";
		return true;
	}
	else{
		cout<<"The both are different\n";
		return false;
	}
}

void main(){
	opOverload a1, a2;
	a1= =a2;
}
```

Answer:

	Runtime Error: Stack Overflow

Explanation:

	Just like normal functions, operator functions can be called recursively. This program just illustrates that point, by calling the operator == function recursively, leading to an infinite loop.

11.

```
class complex{
```

```
        double re;
        double im;
public:
        complex() : re(1),im(0.5) {}
        bool operator==(complex &rhs);
        operator int(){}
};

bool complex::operator == (complex &rhs){
        if((this->re == rhs.re) && (this->im == rhs.im))
                return true;
        else
                return false;
}

int main(){
        complex  c1;
        cout<<  c1;
}
```

Answer : Garbage value

Explanation:

The programmer wishes to print the complex object using output re-direction operator,which he has not defined for his lass.But the compiler instead of giving an error sees the conversion function and converts the user defined object to standard object and prints some garbage value.

12.

```
class complex{
        double re;
        double im;
```

```
public:
        complex() : re(0),im(0) {}
        complex(double n) { re=n,im=n;};
        complex(int m,int n) { re=m,im=n;}
        void print() { cout<<re; cout<<im;}
};

void main(){
        complex c3;
        double i=5;
        c3 = i;
        c3.print();
}
```

Answer:

5,5

Explanation:

Though no operator= function taking complex, double is defined, the double on the rhs is converted into a temporary object using the single argument constructor taking double and assigned to the lvalue.

13.

```
void main()
{
        int a, *pa, &ra;
        pa = &a;
        ra = a;
        cout <<"a="<<a <<"*pa="<<*pa <<"ra"<<ra ;
}
```

Answer :

Compiler Error: 'ra',reference must be initialized

Explanation :

Pointers are different from references. One of the main differences is that the pointers can be both initialized and assigned, whereas references can only be initialized. So this code issues an error.