**80. Explain what we mean by an ACID transaction.**

An ACID transaction is one that is atomic, consistent, isolated, and durable. Durable means that database changes are permanent. Consistency can mean either statement level or transaction level consistency. With transaction level consistency, a transaction may not see its own changes.Atomic means it is performed as a unit.

* **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
* **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
* **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
* **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.
* What is a dangling pointer?
* A pointer initially holding valid address, but later the held address is released or freed. Then such a pointer is called as dangling pointer.
* Can a program be compiled without main() function?
* Yes, it can be but cannot be executed, as the execution requires main() function definition.
* What is the difference between actual and formal parameters?
* The parameters sent to the function at calling end are called as actual parameters while at the receiving of the function definition called as formal parameters.

What are the different ways of passing parameters to the functions? Which to use when?

* **Call by value** − We send only values to the function as parameters. We choose this if we do not want the actual parameters to be modified with formal parameters but just used.
* **Call by reference** − We send address of the actual parameters instead of values. We choose this if we do want the actual parameters to be modified with formal parameters.
* What is typecasting?
* Typecasting is a way to convert a variable/constant from one type to another type.
* **What are local static variables? What is their use?**  
  **Ans:**A local static variable is a variable whose lifetime doesn’t end with a function call where it is declared. It extends for the lifetime of complete program. All calls to the function share the same copy of local static variables. Static variables can be used to count the number of times a function is called. Also, static variables get the default value as 0. For example, the following program prints “0 1”

|  |
| --- |
| #include <stdio.h>  void fun()  {      // static variables get the default value as 0.      static int x;      printf("%d ", x);      x = x + 1;  }    int main()  {      fun();      fun();      return 0;  }  // Output: 0 1 |

Unlike global functions in C, access to static functions is restricted to the file where they are declared. Therefore, when we want to restrict access to functions, we make them static. Another reason for making functions static can be reuse of the same function name in other files.

he **abstract class** will hold common functionality for all **classes** that extend it

**What are the different levels of abstraction in the DBMS?**

**Ans:** There are 3 levels of data abstraction in the DBMS.

**They include:**

* **Physical Level:**This is the lowest level of the data abstraction which states how the data is stored in the database.
* **Logical Level:**This is the next level of the data abstraction which states the type of the data and the relationship among the data that is stored in the database.
* **View Level:** This is the highest level in the data abstraction which shows/states only a part of the database

**#16) What integrity rules exist in the DBMS?**

**Ans:** There are 2 major integrity rules that exist in the DBMS.

**They are:**

* **Entity Integrity:** This states a very important rule that value of a Primary key can never have a NULL value.
* **Referential Integrity:**This rule is related to the Foreign key which states that either the value of a Foreign key is a NULL value or it should be the primary key of any other relation.