**Data structure interview question**: What is data structure**?**  
**Answer**: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.  
 **Data structure interview question**:List out the areas in which data structures are applied extensively?

**Answer**:

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

***Q. What are the major data structures used in the following areas of RDBMS, Network data model & Hierarchical data model.***

**Answer**: Ø RDBMS – Array (i.e. Array of structures)  
Ø Network data model – Graph  
Ø Hierarchical data model – Trees  
  
***Q:*** ***If you are using C language to implement the heterogeneous linked list, what pointer type will you use?***

**Answer**: 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.  
  
***Q: Minimum number of queues needed to implement the priority queue?*  
Answer**: Two. One queue is used for actual storing of data and another for storing priorities.

***Q:What is the data structures used to perform recursion?*Answer:** 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.

***Q:What do you mean by: Syntax Error, Logical Error, Run time Error?***

**Answer**:

**Syntax Error**-Syntax Error is due to lack of knowledge in a specific language. It is due to somebody does not know how to use the features of a language.We can know the errors at the time of compilation.logical

**Error**-It is due to the poor understanding of the requirement or problem.

**Run time Error**-The exceptions like divide a number by 0,overflow and underflow comes under this.  
  
**Data structure interview question** :What is the maximum total number of nodes in a tree that has N levels? Note that the root is level (zero).

**Answer:**2^(N+1)-1..  
  
if N=0; it is 2-1=1,1 is the max no of node in the tree   
if N=1; it is 4-1=3, 3 is the max no of nodes in the tree  
if N=2; it is 8-1=7, 7 is the max.  
  
**Data structure interview question**:Explain about the types of linked lists?

**Answer**:Their are three linked lists1)linear or simple linked lists2)doubly linked lists3)circular linked lists  
**simple linked list :**This contains a node which has two parts, see that a node is a STRUCTURE.one is data and other one is a pointer which is called self reference pointers, so we must make it to point to the next location of second node created dynamically.  
**double linked lists :**A node will consists of previous node address , a data & next node address which can move backwards to the very first address.  
**circular linked list** **:**Here we will have the node consists of same thing but default when it finishes the last node and come to the first node.  
  
**Data structure interview question**:What are the parts of root node?   
**Answer**:A root node contains data part and has link part. i.e links to its child. If it is binary tree it has two links i.e left child and right child.  
  
**Data structure interview question:**How will inorder, preorder and postorder traversals printthe elements of a tree?  
**Answer**:   
voidinorder(node \* tree)   
{  
 if(tree!=NULL)  
 {   
inorder(tree->leftchild);   
printf("%d",tree->data);   
inorder(tree->rightchild);  
 }   
else  
return;   
}   
void postorder(node \* tree)   
{   
if(tree!=NULL)   
{   
postorder(tree->leftchild);   
postorder(tree->rightchild);   
printf("%d",tree->data);  
 }   
else  
return;   
}   
void preorder(node \* tree)  
 {   
if(tree!=NULL)   
{   
printf("%d",tree->data);   
preorder(tree->leftchild);   
preorder(tree->rightchild);   
}   
else   
return;   
}