Data Structure QB Answers:

ARRAY::REMEMBER

1. **What is the importance of data structure in computer science?**

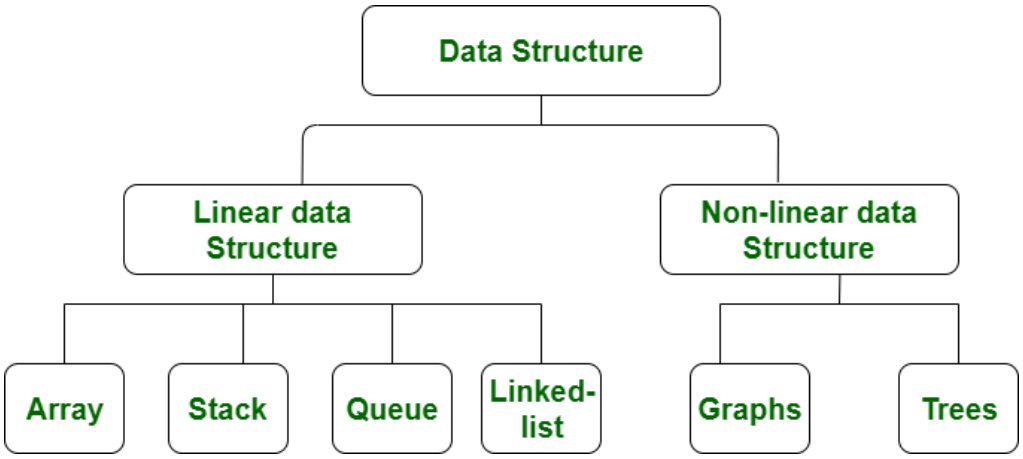
Ans: Data structure and algorithms are two of the most important aspects of computer science. Data structures allow us to organize and store data, while algorithms allow us to process that data in a meaningful way. Learning data structure and algorithms will help us become a better programmer.

2. **Differentiate data structure, algorithm and program with a suitable example.**

Ans: Data structures hold data in convenient ways for processing. An algorithm is an organized way and collection of steps to solve problems; Programs join these things together in an organized, procedural way to produce the required output.

3. **What are the types of data structure? What are the need and applications of data structure? What do you mean by Dynamic Data structure?**

Ans:



Need for Data Structures: Data structures bring together the data elements in a logical way and facilitate the effective use, persistence and sharing of data. They provide a formal model that describes the way the data elements are organized. Data structures are the building blocks for more sophisticated applications.

Dynamic Data Structures: A dynamic data structure (DDS) refers to an organization or collection of data in memory that has the flexibility to grow or shrink in size, enabling a programmer to control exactly how much memory is utilized. Dynamic data structures change in size by having unused memory allocated or de-allocated from the heap as needed.

4. **What is Abstract Data type? Explain Abstract Data type (ADT) with examples.**

Ans: An Abstract Data Type in a data structure is a kind of data type whose behaviour is defined with the help of some attributes and some functions. Generally, we write these attributes and functions inside a class or a structure so that we can use an object of the class to use that particular abstract data type. E.g. List, Stack, Queue

5. **Can a primitive data type be considered as an ADT?**
6. **List out some of the areas in which data structures are used?**

Ans: Data Structures are used in:

- Data structures are used in any program or software.
- They are used in the areas of
- Compiler Design
- Operating System
- DBMS
- Graphics
- Simulation
- Numerical Analysis
- Artificial Intelligence

7. **What are the major data structures used in the following areas: RDBMS, Network data model and Hierarchical data model?**

Ans: In Relational Database Management System, the data structures used are Arrays of structures.

In Network data Model, the Graph Data Structure is used.

In Hierarchical data model, the Tree Data Structure is used.

8. **Define an algorithm. What are the properties of an algorithm?**

Ans: Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output. Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

Properties of algorithms:

- **Unambiguous** – Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- **Input** – an algorithm should have 0 or more well-defined inputs.
- **Output** – an algorithm should have 1 or more well-defined outputs, and should match the desired output.
- **Finiteness** – Algorithms must terminate after a finite number of steps.
- **Feasibility** – should be feasible with the available resources.
- **Independent** – an algorithm should have step-by-step directions, which should be independent of any programming code.

9. **Which properties of an algorithm distinguish from program?**

Ans: The main difference is between algorithm and program is that an algorithm is a step-by-step procedure for solving the problem while programming is a set of instructions for a computer to follow to perform a task. A program could also be an implementation of code to instruct a computer on how to execute an algorithm.

10. **What is the Brute Force algorithm?**

Ans: A brute force approach is an approach that finds all the possible solutions to find a satisfactory solution to a given problem. The brute force algorithm tries out all the

possibilities till a satisfactory solution is not found. It is the straight-forward way to approach a problem.

11. What values are automatically assigned to those array elements which are not explicitly initialized?

Ans: In C/C++, a non-initialized array will assign garbage values to itself. A pointer will assign the NULL value and a Boolean array will assign false value by default.

In JAVA, any non-initialized variable assigns 0 to itself.

12. What are the limitations of arrays? How can you overcome the limitations of arrays? In spite of this limitation why it is important?

Ans: The limitations of an array are explained below –

- An array which is formed will be homogeneous. That is, in an integer array only integer values can be stored, while in a float array only floating value and character array can have only characters. Thus, no array can have values of two data types.
- While declaring an array, passing size of an array is compulsory, and the size must be a constant. Thus, there is either shortage or wastage of memory.
- Shifting is required for insertion or deletion of elements in an array.
- An array doesn't check boundaries: In C language, we cannot check, if the values entered in an array are exceeding the size of that array or not.
- Data that is entered with the subscript, exceeds the array size and will be placed outside the array. Generally, on the top of the data or the program itself. This will lead to unpredictable results, to say the least. Also, there will be no error message to warn the programmer of going beyond the array size. In some cases, the program may hang.

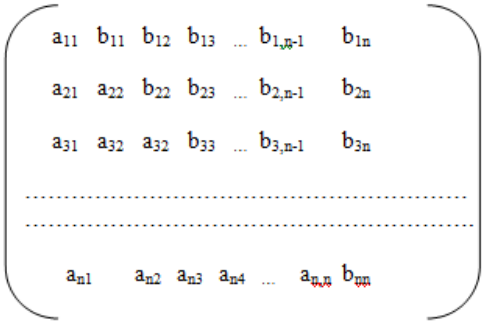
All the above limitations except the first one can be overcome by using Linked-Lists.

In spite of the above limitations, arrays are very useful in indexing and pointing to any desired element in constant time complexity.

13. Explain an efficient way of storing two symmetric matrices of the same order in memory.

Ans: An n-square matrix array will be symmetric if a[j][k] = a[k][j] for all j and k.

For a symmetric matrix, we need to store elements which are lying on and below the diagonal of the matrix or those on and above the diagonal. Two symmetric matrix of A and B of the same dimension can be stored in an n*(n+1) array C where c[j][k]=a[j][k] when j ≥ k but c[j][k-1] = b[j][k-1] when j < k.



14. **How data can be organized in memory? What is static and dynamic memory allocation? Explain with example.**

Ans: The Memory is divided into three sections.

Heap Memory: It is a part of the main memory. It is unorganized and treated as a resource when you require the use of it. Heap memory can't be used directly with the help of a pointer.

Stack Memory: It stores temporary variables created by a function. In stack, variables are declared, stored, and initialized during runtime. It follows the First in last out method that means whatever element is going to store last is going to delete first when it's not in use.

Code Section: Whenever the program is executed it will be brought into the main memory. This program will get stored under the code section. Based upon the program it will decide whether to utilize the stack or heap sections.

Static Memory Allocation: Static Memory is allocated for declared variables by the compiler. The address can be found using the address of (& in C/C++) operator and can be assigned to a pointer. The memory is allocated during compile time. In static memory allocation whenever the program executes it fixes the size that the program is going to take, and it can't be changed further. So, the exact memory requirements must be known before.

Dynamic Memory Allocation: Memory allocation done at the time of execution (run time) is known as dynamic memory allocation. Functions calloc(), malloc() and realloc() support allocating dynamic memory. In the Dynamic allocation of memory space is allocated by using these functions when the value is returned by functions and assigned to pointer variables.

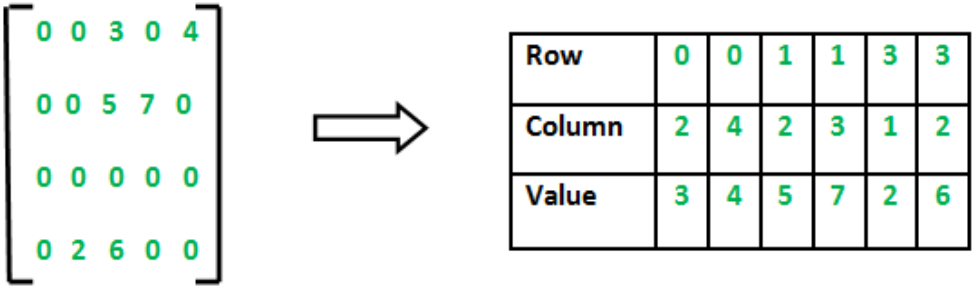
E.g. int *n=(int *)malloc(Num*sizeof(int))

We can also allocate memory dynamically using the new keyword in C++ and JAVA.

15. **What is a sparse matrix? How is it represented in memory? What are the types of sparse matrices?**

Ans: A 2-dimentional array with most of its value being equal to 0 is a sparse matrix.

Representing a sparse matrix by a 2D array leads to wastage of lots of memory as zeroes in the matrix are of no use in most of the cases. So, instead of storing zeroes with non-zero elements, we only store non-zero elements. This means storing non-zero elements with triples- (Row, Column, value).



Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

Types of sparse matrices:

- Non-Regular Sparse Matrices:
- Regular Sparse Matrices
    - Lower Triangular Sparse Matrix.
    - Upper Triangular Sparse Matrix.
    - Tri-Diagonal Regular Sparse Matrix.

16. What are different characteristics of an algorithm? Explain briefly.

Ans: Different Characteristics are:

- Unambiguous – Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- Input – An algorithm should have 0 or more well-defined inputs.
- Output – An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- Finiteness – Algorithms must terminate after a finite number of steps.
- Feasibility – Should be feasible with the available resources.
- Independent – An algorithm should have step-by-step directions, which should be independent of any programming code.

17. What is subscripted variable? Differentiate between Primitive Data Structure and Primitive Data-Type.

Ans: **Subscripted variables** are used to store the values of the same type in an array. It also helps us to store more values in a single variable name. Subscripted variables are declared by giving the variable name along with the subscript.

Primitive Data-Type	Primitive Data Structure
A primitive data type is either a data type that is built into a programming language, or one that could be characterized as a basic structure for building more sophisticated data types.	Primitive data structure is a fundamental type of data structure that stores the data of only one type.

18. What does the following function do for a given Linked List with first node as head?

```
1 void fun1(struct node* head){
2         if(head == NULL)
3                   return;
4 fun1(head->next);
5 printf("%d ", head->data);
6 }
```

Ans: The above code block displays the elements of the linked list in reverse order.

ARRAY::UNDERSTAND

1. **Write an algorithm to find the smallest element in the array.**

Ans: Algorithm:

1. Input the array elements.
2. Initialize small = arr[0]
3. Repeat from i = 1 to n-1
4. if(arr[i] < small)
5. small = arr[i]
6. Print small.

2. **What is the addressing formula for an element A[i][j] in row major order, if i and j are bounded by the lower and upper limits as l1 ≤ i ≤ u1 and l2 ≤ j ≤ u2? [Assume that the base address is L and w be the number of words allocated to each element]**

Ans: <u>For Row-Major Order:</u>

Address of A[I][J] = B + W * ((I – LR) * N + (J – LC))
I = Row Subset of an element whose address to be found,
J = Column Subset of an element whose address to be found,
B = Base address,
W = Storage size of one element store in an array(in byte),
LR = Lower Limit of row/start row index of the matrix(If not given assume it as zero),
LC = Lower Limit of column/start column index of the matrix(If not given assume it as zero),
N = Number of column given in the matrix.

<u>For Colum-Major Order:</u>

Address of A[I][J] = B + W * ((J – LC) * M + (I – LR))
I = Row Subset of an element whose address to be found,
J = Column Subset of an element whose address to be found,
B = Base address,
W = Storage size of one element store in any array(in byte),
LR = Lower Limit of row/start row index of matrix(If not given assume it as zero),
LC = Lower Limit of column/start column index of matrix(If not given assume it as zero),
M = Number of rows given in the matrix.

3. **Write an algorithm to multiply two two-dimensional array of size m × n.**

Ans: matrixMultiply(A, B):

Assume dimension of A is (m x n), dimension of B is (p x q)

Begin

if n is not same as p, then exit

otherwise define C matrix as (m x q)

for i in range 0 to m - 1, do

```
    for j in range 0 to q – 1, do
      for k in range 0 to p, do
        C[i, j] = C[i, j] + (A[i, k] * A[k, j])
      done
    done
  done
End
```