FDS Question bank : Insem

UNIT 1

Q1. Explain the relationship between ADT and data structure.

Ans. The relationship between an Abstract Data Type (ADT) and a data structure is one of abstraction and implementation.

1. **Abstract Data Type (ADT):**
    - An ADT is a high-level representation of a data structure's behavior and operations without showing the implementation details.
    - It defines a set of operations or methods that can be performed on the data structure, along with their semantics and constraints.
    - ADTs focus on **'what needs to be done with the data structure, not how it's done'.**
2. **Data Structure:**
    - A data structure is the concrete implementation of an ADT. It is a way of organizing and storing data in memory to support the operations defined by the ADT.
    - Data structures specify how data is stored, accessed, and manipulated in detail.
    - Examples of data structures include arrays, linked lists, stacks, queues, trees, and graphs.

Q.2 Write an algorithm to find smallest number among N numbers and determine the efficiency.

Ans.

```
1. Initialize a variable "smallest" to positive infinity (or a very large number).

2. Read the value of N (the number of elements.

3. For i = 1 to N:

   a. Read the next number "num".

   b. If "num" is less than "smallest," update "smallest" to "num".

4. Print the value of "smallest" as the smallest number.
```

Algorithm Efficiency:

- Time Complexity: O(N) - It requires exactly N comparisons to find the smallest number.

- Space Complexity: O(1) - It uses a constant amount of extra space (for the "smallest" variable).

Q.3 Write an algorithm to search a key, among N numbers and determine the efficiency

Ans.

```
1. Read the value of N (the number of elements).

2. Read the key to search for.

3. Initialize a variable "found" to false.

4. For i = 1 to N:

    a. Read the next number "num".

    b. If "num" is equal to the "key," set "found" to true and break out of the loop.

5. If "found" is true, print "Key found."

   Otherwise, print "Key not found."
```

Algorithm Efficiency:

- Time Complexity: O(N) - In the worst case, the algorithm may have to iterate through all N numbers to find the key or determine that it's not present.

- Space Complexity: O(1) - It uses a constant amount of extra space (for the "found" variable).

Q.4 What is an algorithm? write the essential properties characteristics of a good algorithm.

Ans.

Def : An algorithm is a step by step procedure that defines a set of instructions that must be carried out in a specific order to produce the desired result

Algorithms are generally developed independently of underlying languages.

<u>Characteristics of an algorithm :</u>

Clear and unambiguous :The algorithm should be distinct in every way and lead to a single conclusion.

Well defined inputs : The algorithm must indicate what output will be produced, as well as be well defined

well defined outputs : The algorithm must clearly indicate what output will be produced, as well as well defined

Finiteness : The algorithm must be finite. It must not result in infinite loops as similar situations.

feasible : The algorithm should be simple. Generic and practical, and it is able to be executed with the resources available. It can’t contend any futuristic technology or anything like that.

language independent : The design algorithm must be simple instructions in readable manner.

Q5. Find time complexity of the following code

```
 : For i=1 to n do        ----------(n)

For j=i+1 to n do         ----------(n-1)

For k=j+1 to n do         ----------(n-2)

X = x + 1                           ----------n^3

Finally answer is :       O(n^3)
```

5. What is software,Program, give the need of it. [5 Marks]

Ans. Software:

Software refers to a collection of computer programs, data, and instructions that tell a computer how to perform specific tasks or functions. In simpler terms, software is the intangible component of a computer system that enables it to carry out various operations, from running applications to managing hardware resources.

Software can be categorized into two main types:

1. **System Software**: This type of software includes the operating system, device drivers, and utilities that facilitate the interaction between the hardware and application software. It manages computer hardware resources and provides a foundation for running other software.

2. **Application Software**: Application software includes all the programs and tools designed for specific tasks or applications, such as word processors, web browsers, video games, and business software like spreadsheets and databases.

Program: It is a set of instructions written in a programming language that a computer can understand and execute. Programs define the logic and functionality of software applications. They consist of sequences of commands, statements, and algorithms that instruct the computer on how to perform a particular task or solve a specific problem.

Programs can be categorized into various types, including:

1. **Application Programs**: These are end-user programs designed for specific tasks, such as word processing, web browsing, or playing games.

2. **System Programs**: These programs assist in the management and control of computer hardware and resources, such as operating systems, device drivers, and utility programs.

3. **Scripting Programs**: Scripts are sets of instructions written in scripting languages to automate tasks or customize the behavior of software applications.

Needs of Software: The need for software in modern computing is driven by several essential factors:

1. **Automation**: Software enables automation of tasks, reducing the need for manual intervention and increasing efficiency. Businesses use software to automate processes, such as payroll, inventory management, and customer relationship management.

2. **Problem Solving**: Software allows computers to solve complex problems that would be impractical or impossible for humans to tackle manually. This includes tasks in science, engineering, data analysis, and simulations.

3. **User Interaction**: Software provides a user-friendly interface for individuals to interact with computers. This includes graphical user interfaces (GUIs) and command-line interfaces (CLIs) that allow users to run applications, manage files, and perform various tasks.

4. **Data Processing**: Software is crucial for data processing, storage, and retrieval. Databases, data analysis tools, and data management software are essential for businesses and organizations to make informed decisions.

5. **Entertainment and Creativity**: Software is responsible for creating video games, multimedia content, art, music, and other forms of entertainment and creative expression.

6. **Communication**: Software enables communication through email clients, messaging apps, social media platforms, and video conferencing tools, connecting people worldwide.

7. **Security**: Software plays a critical role in computer security. Antivirus software, firewalls, and encryption tools protect against cyber threats and safeguard sensitive data.

8. **Customization**: Businesses and individuals can use software to tailor their computing experience to their specific needs and preferences.

6. Which are the different data types available in programming languages you know, Explain it.

Ans. Integer (int): Integers are whole numbers without any fractional parts. They can be positive or negative. Examples: -3, 0, 42.

1. **Floating-Point** (**float or double**): Floating-point numbers represent real numbers with decimal points. They can store both whole and fractional parts. Examples: 3.14, -0.5, 2.71828.

2. **Character** (char): Character data types represent individual characters, such as letters, digits, or symbols. Examples: 'A', '7', '$'.

3. **Boolean** (bool): Boolean data types have two possible values: true or false. They are often used for conditional expressions and logical operations. Examples: true, false.

4. **String**: Strings are sequences of characters. They allow you to store and manipulate text. Examples: "Hello, World!", "12345".

5. **Array**: Arrays are collections of elements of the same data type. They can store multiple values of the same kind, accessed by their index. Examples: [1, 2, 3, 4, 5], ["apple", "banana", "cherry"].

6. **Struct (or Record):** A struct is a composite data type that groups together variables of different data types under a single name. It's used for creating custom data structures. Examples: struct Person { string name; int age; }.

7. **Pointer**: Pointers store memory addresses, allowing you to work with memory directly. They are often used in low-level programming for tasks like dynamic memory allocation. Example: int* ptr; (a pointer to an integer).

8. **Enumerated (enum):** An enumerated data type defines a set of named integer values. It's useful for creating named constants. Example: enum Days { Monday, Tuesday, Wednesday, Thursday, Friday }.

9. **Date and Time**: Some programming languages offer data types specifically for working with dates and times, allowing you to manipulate and format dates and perform date calculations.

10. **Custom/User-Defined Types**: Many languages allow you to define your own data types using classes, structures, or type aliases. These custom types can encapsulate complex data and behaviors.

11. **Nullable Types**: Some languages support nullable data types, which can hold a regular value or a special "null" value to represent the absence of data.
12. **Big Numbers**: Some languages provide data types for handling arbitrarily large numbers, which are not limited by the size of native integer or floating-point types.
13. **Complex Numbers**: In scientific and mathematical programming, complex number data types are available for working with complex numbers (numbers with real and imaginary parts).
14. **Decimal**: A decimal data type is used for high-precision arithmetic and is often used in financial and monetary calculations to avoid rounding errors.

7. Explain Persistent and ephemeral data structures, static and dynamic data structure

Ans.

Persistent : A data structure that supports operations on the most recent version as well as the previous versions, is term as the persistent data structure. A persistent data structure is partially persistent. If any version can be accessed by only the most recent 1 can be updated, It is fully persistent if any version can be both accessed and updated. They allow you to perform operations on a data structure while preserving previous versions, enabling you to access the data's history.

ephemeral : An epimeral data structure is one that supports operations only on the most recent version.They are designed to store and manipulate data in the current state of a program, and they do not retain any history of previous versions or changes.

static : static data strucutures are designed to have a fixed size or capacity that cannot be changed once they are created. They are usually used when we know a specific size of a data structure. It is created before the program execution. An array is static data structure.

Dynamic : Data structure that is created at runtime is called dynamic data structure. The variable of this type are not always referenced by the user defined name. These are access indirectly, using their addresses through pointers. A linked list is a dynamic data structure. When realized using dynamic memory management and pointers, whereas an array is a static data structure trees and graphs can be implemented as dynamic data structure.

8. **What is Abstract Data Type? Write an ADT for rational number**

Ans. An abstract data type is one in which the set of operations is defined as a formal logical level, without being restricted by the operational details an ADT includes declaration of data, implementation of operations and encapsulation of data and operations Consider the concept of the queue, at least three data structures will supports the queue.

Abstract Data Type, is a high-level description of a data structure that specifies a set of operations that can be performed on the data and the properties or constraints of those operations, without specifying the implementation details. ADTs are used to abstract and encapsulate data, making it easier to reason about and manipulate data in a program.

1. A rational number is a number that can be expressed as a fraction, where both the numerator and the denominator are integers. Rational numbers can be positive or negative and can represent exact values such as 1/2 or 3/4.

CreateRational(numerator: Integer, denominator: Integer) -> RationalNumber:

- Description: Creates a rational number object with the given numerator and denominator.
- Precondition: The denominator must not be zero.
- Postcondition: Returns a rational number object representing **numerator / denominator**.

2. **Add(rational1: RationalNumber, rational2: RationalNumber) -> RationalNumber**:
    - Description: Adds two rational numbers and returns the result as a new rational number.
    - Precondition: None.
    - Postcondition: Returns a rational number representing the sum of **rational1** and **rational2**.
3. **Subtract(rational1: RationalNumber, rational2: RationalNumber) -> RationalNumber**:
    - Description: Subtracts **rational2** from **rational1** and returns the result as a new rational number.
    - Precondition: None.
    - Postcondition: Returns a rational number representing the difference of **rational1** and **rational2**.
4. **Multiply(rational1: RationalNumber, rational2: RationalNumber) -> RationalNumber**:
    - Description: Multiplies two rational numbers and returns the result as a new rational number.
    - Precondition: None.
    - Postcondition: Returns a rational number representing the product of **rational1** and **rational2**.

5. **Divide(rational1: RationalNumber, rational2: RationalNumber) -> RationalNumber**:

   - Description: Divides **rational1** by **rational2** and returns the result as a new rational number.
   - Precondition: **rational2** must not be zero.
   - Postcondition: Returns a rational number representing the division of **rational1** by **rational2**.

6. **Equals(rational1: RationalNumber, rational2: RationalNumber) -> Boolean**:

   - Description: Compares two rational numbers for equality.
   - Precondition: None.
   - Postcondition: Returns **true** if **rational1** is equal to **rational2**, otherwise returns **false**.

7. **ToString(rational: RationalNumber) -> String**:

   - Description: Converts a rational number to a string representation.
   - Precondition: None.
   - Postcondition: Returns a string representation of the rational number.

Q9.

Find order of magnitude to compute N^M
1) Read n,m : simply read the value of n and m which do not contribute to anything
2) Let r=1 : initializing the single varieable to a constant value which takes constant time
3) For I=1 to M do : loops runs M times , where M is a exponent
4) R=R*n : single multiplication operation which executes M times
5) Print R : printing the required answer
6) Stop

At last to find the order of magnitude to computer n^m using this algorithm, we can say it is O(M), which is its time complexity, Because the number of iterations in the loop is determined by the value of M.

10. Obtain Frequency count for the following code (Marks 8 May 12)

```
for(i=1; i<=n; i++)

        for(j=1; j<=n; j++)
```

```
{
        c[i][j]=0;
        for(k=1;k<=n;k++)
        {
            C[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
}
```

UNIT 2

1. What is a sparse matrix ? Explain with *one* example

Ans. Sparse matrix :

A sparse matrix is a matrix in which most of the elements are zero. In other words, it is a matrix in which the number of non-zero elements is much smaller than the number of zero elements.

In sparse matrix we can store the all non zero elements in rows and columns, with proper indices.

0 0 3 0

4 0 0 5

7 0 0 0

0 0 0 2

This matrix has 16 elements, but only 7 of them are non-zero. Therefore, it is a sparse matrix.

Sparse matrices are often used in scientific computing and machine learning. For example, they can be used to represent graphs, images, and other types of data that contain many zero elements.

Sparse matrices can be represented in a variety of ways, but one common way is to use a triplet representation. In a triplet representation, each non-zero element is represented by a triple consisting of its row index, column index, and value.

Representation of sparse to ideal matrix is as follow :

(0, 2, 3)

(1, 3, 4)

(2, 0, 7)

(2, 3, 5)

(3, 2, 2)

2. Write an algorithm for fast transpose of sparse matrix and find out its time complexity.

Ans.

3. Show how a two-dimensional array is stored in memory. Assume that array start at the address 4000.

4.Explain how a polynomial is represented using an array with *one* example.

5. Write the algorithm for sparse matrix addition

6. Consider a Matrix of 3*3 [10, 0, 0, 21,19,0,45,28,15]

a. convert it to a sparse triplet representation

b. convert it to a Transpose by simple transpose algorithm

c. write a algorithm of simple transpose

7. Explain how a polynomial is represented using an array with the following example.
4x^5+10x^3-30. Write a CPP program to implement a polynomial using an array and perform its evaluation.

8. Write a ‘C++’ program to implement polynomial using array and perform its multiplication

9. What is Abstract Data Type (ADT) ? Write an abstract data type for Arrays

10.Define the following terms with example:

i) Data object

ii) Data structure

iii) Abstract Data Type

iv) Explain Big-oh,

11.Explain the two-dimensional array in detail with column and row major representation and address calculation in both the cases.

12.Explain polynomial representation of an array and also write data structure declaration with suitable example.

13.Explain concept Generalized Linked List and representation polynomial using array with given example:

4x3+2x2+6xy+7xy3.

14. Explain concept of arrays with suitable examples.

15.What is the need of fast transpose algorithm?

15. Write pseudo C/C++ code to perform simple transpose of sparse matrix.

16. State the different characteristics of an algorithm.

17. What is sparse matrix ? Explain its representation with an example.

18.Explain the need for fast transpose of sparse matrix. Comment on its time complexity.

19. Give pseudo C/ C++ code to concatenate two strings.

20. Define and explain the following terms :

(a) Linear data structure

(b) Non-linear data structure

(c) Time complexity

(d) Space complexity

21. Explain fast transpose of sparse transpose with examples.

22. Derive address calculation formula one dimensional array with one example.

23. Difference between array & ordered list.

24. Multidimensional array & their address calculation with example.

25. Write & explain fast transpose algorithm of sparse matrix.

26. Give pseudo C/ C++ code to concatenate two strings.