**Task 1:**

What do you understand about data structures?

Data structures are organized ways to store and manage data in computer programs. Just like organizing items in containers, data structures help us arrange information so we can easily access and modify it when needed. They are fundamental building blocks in programming that determine how data is stored in computer's memory, making it efficient to perform operations like adding, removing, or searching for information. The choice of data structure affects how quickly and efficiently a program can perform these operations, which is crucial for writing effective programs. Understanding data structures is essential because they help solve complex problems by organizing data in ways that make sense for specific tasks.

**Task 2:**

What are the types of data structures you know .. list them out..

* Arrays
* LinkedList
* Stack
* Queue
* Trees
* Graphs
* Hash Tables
* Hash maps
* Array List
* Dynamic Arrays

**Task 3:**

What all operations can we do in Data structures?

* Insertion
* Deletion
* Traversal
* Searching
* Sorting
* Merging

**Task 4:**

What are static and dynamic arrays? Explain or summarize key points in a table like

Size, performance, memory, flexibility, limitations

|  |  |  |
| --- | --- | --- |
|  | **Static Arrays** | **Dynamic Arrays** |
| **Size** | Fixed size that can’t change | Can grow or shrink as needed |
| **Performance** | Very fast | Slightly slower when growing |
| **Memory** | Uses exact space needed | Uses some extra space |
| **Flexibility** | Cannot be changed once created | Can be changed anytime |
| **Limitations** | Must know size beforehand | Uses more computer memory |

**Task 5:**

What is the binary value of a?

Hint ascii value is 97..

ASCII value of 'a' = 97

Converting 97 to Binary:

97/2 = 48 remainder 1

48/2 = 24 remainder 0

24/2 = 12 remainder 0

12/2 = 6 remainder 0

6/2 = 3 remainder 0

3/2 = 1 remainder 1

1/2 = 0 remainder 1

Binary = 1100001

Character: 'a'

ASCII: 97

Binary: 01100001

**Task 6:**

Types of Computer memory with examples.. Explain ..

1. Primary Memory

Volatile Memory:

RAM (Random Access Memory)

- DRAM (Dynamic RAM): Main system memory, needs constant refreshing

- SRAM (Static RAM): Faster than DRAM, used in cache memory

Non-Volatile Memory:

ROM (Read-Only Memory)

- BIOS/UEFI ROM: Stores boot firmware

- EPROM/EEPROM: Programmable/erasable ROM

- Flash Memory: Modern non-volatile storage

2. Cache Memory

CPU Cache - Ultra-fast memory close to processor

- L1 Cache: Fastest, smallest (32-64KB per core)

- L2 Cache\*\*: Moderate speed (256KB-1MB per core)

- L3 Cache: Shared cache (8-32MB)

3. Secondary Storage

Magnetic Storage

- Examples: Hard Disk Drives (HDDs), tape drives

Solid State Storage\*\*

- Examples: SSDs, NVMe drives, eMMC

Optical Storage - Examples: CDs, DVDs, Blu-ray discs

**Task 7:**

Reverse an array. write a code.

public class task07 {

    public static void main(String[] args) {

        int[] digits = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};

        System.out.println("Original array:");

        printArray(digits);

        reverseArray(digits);

        System.out.println("Reversed array:");

        printArray(digits);

    }

    public static void reverseArray(int[] arr) {

        int start = 0;

        int end = arr.length - 1;

        while (start < end) {

            int temp = arr[start];

            arr[start] = arr[end];

            arr[end] = temp;

            start++;

            end--;

        }

    }

    public static void printArray(int[] arr) {

        for (int digit : arr) {

            System.out.print(digit + " ");

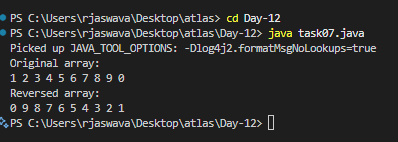
        }

        System.out.println();

    }

}

**Output**



**Task 8:**

Reverse a string .. write a code.

public class task08 {

    public static void main(String[] args) {

        String original = "jaswanth";

        System.out.println("Original string: " + original);

        String reversed = reverseString(original);

        System.out.println("Reversed string: " + reversed);

    }

    public static String reverseString(String str) {

        char[] charArray = str.toCharArray();

        int start = 0;

        int end = charArray.length - 1;

        while (start < end) {

            char temp = charArray[start];

            charArray[start] = charArray[end];

            charArray[end] = temp;

            start++;

            end--;

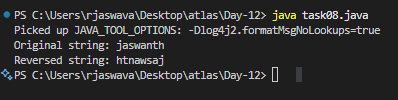
        }

        return new String(charArray);

    }

}

**Output**



**Task 09:**

import java.util.Scanner;

public class task09 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a name: ");

        String name = scanner.nextLine();

        System.out.println("Original name: " + name);

        String reversed = reverseString(name);

        System.out.println("Reversed name: " + reversed);

        scanner.close();

    }

    public static String reverseString(String str) {

        char[] charArray = str.toCharArray();

        int start = 0;

        int end = charArray.length - 1;

        while (start < end) {

            char temp = charArray[start];

            charArray[start] = charArray[end];

            charArray[end] = temp;

            start++;

            end--;

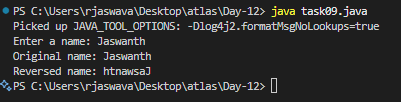
        }

        return new String(charArray);

    }

}

**Output**



**Task 10:**

public class task10 {

   public static void main (String[] args) {

      int[] arr1 = {11, 34, 66, 75};

      int n1 = arr1.length;

      int[] arr2 = {1, 5, 19, 50, 89, 100};

      int n2 = arr2.length;

      int[] merge = new int[n1 + n2];

      int i = 0, j = 0, k = 0, x;

      System.out.print("Array 1: ");

      for (x = 0; x < n1; x++)

      System.out.print(arr1[x] + " ");

      System.out.print("\nArray 2: ");

      for (x = 0; x < n2; x++)

      System.out.print(arr2[x] + " ");

      while (i < n1 && j < n2) {

         if (arr1[i] < arr2[j])

            merge[k++] = arr1[i++];

         else

            merge[k++] = arr2[j++];

      }

      while (i < n1)

      merge[k++] = arr1[i++];

      while (j < n2)

      merge[k++] = arr2[j++];

      System.out.print("\nArray after merging: ");

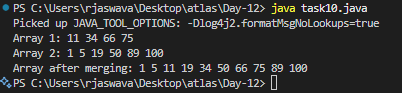
      for (x = 0; x < n1 + n2; x++)

      System.out.print(merge[x] + " ");

   }

}

**Output:**



**Explanation:**

This code merges two arrays into single big array

**Task 11:**

What do you understand by Hash table?

A Hash Table is a data structure that stores information using a key-value system. It uses a special function called a hash function that converts keys into storage locations. When data needs to be stored, the hash function creates a unique address where the value will be placed. This makes finding and retrieving data very fast because instead of searching through all items, the hash table can directly go to the correct storage location. The main purpose of a hash table is to provide quick access to data without having to search through everything. It's efficient because it can find any piece of data in roughly the same amount of time, regardless of how much information is stored. Hash tables are useful when you need to store and retrieve data quickly, and they're designed to handle large amounts of information efficiently.

**Task 12**

Understand the below Hash table code and try to print values using get method of Hash table

import java.util.Hashtable;

import java.util.Map;

public class task12 {

    public static void main(String[] args) {

        Hashtable<String, Integer> ht = new Hashtable<>();

        ht.put("Anitha", 101);

        ht.put("Kavitha", 102);

        ht.put("Meera", 103);

        // use  get method of Ht

        System.out.println("Anitha's value: " + ht.get("Anitha"));

        System.out.println("Kavitha's value: " + ht.get("Kavitha"));

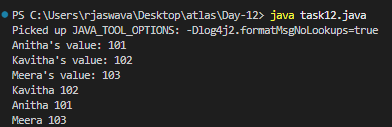
        System.out.println("Meera's value: " + ht.get("Meera"));

        for (Map.Entry<String, Integer> e : ht.entrySet())

            System.out.println(e.getKey() + " " + e.getValue());

    }

}

**Output:** 

**Task 13:**

import java.util.HashMap;

import java.util.Map;

public class task13 {

    public static void main(String[] args) {

        HashMap<String, Integer> hm = new HashMap<>();

        hm.put("Anitha", 101);

        hm.put("Kavitha", 102);

        hm.put("Meera", 103);

        // Using get method

        System.out.println("Anitha's value: " + hm.get("Anitha"));

        System.out.println("Kavitha's value: " + hm.get("Kavitha"));

        System.out.println("Meera's value: " + hm.get("Meera"));

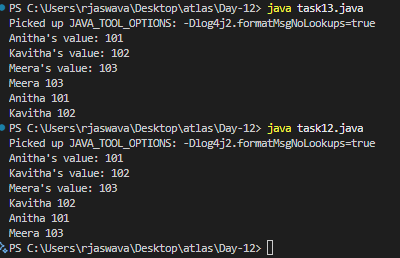
        for (Map.Entry<String, Integer> e : hm.entrySet())

            System.out.println(e.getKey() + " " + e.getValue());

    }

}

**Output:**



Task 14:

Difference between Hash Table and Hash Map

Similarities

hash table and Hash Map have linked list internally.

Collisions occur in Hash Table and hash Maps.

Collision in Hash map can handle separate chaining, Open addressing etc..

1. Null Values

* HashMap allows null key and values
* HashTable doesn’t allow null key or values

1. Synchronization

* HashMap is not synchronized (not thread safe)
* HashTable is synchronized (thread safe)

1. Performance

* HashMap is generally fast
* HashTable is slower due to synchronization

**Task 15:**

Linear probing in Hash table

**Output:**

**Task 16:**

Try to add 1 null value in the key and run the hash map code..

Also add one more null value to the key and see the result..

make a Hashmap synchronized..

Plz note : Hash Maps are - asynchronous in nature..

Hint   Map<String, Integer> syncMap = Collections.synchronizedMap(Hm1);

import java.util.HashMap;

import java.util.Map;

import java.util.Collections;

public class task16 {

    public static void main(String[] args) {

        HashMap<String, Integer> hm = new HashMap<>();

        hm.put("Anitha", 101);

        hm.put("Kavitha", 102);

        hm.put("Meera", 103);

        hm.put(null, 104);

        hm.put(null, 105);

        // Using get method

        System.out.println("Anitha's value: " + hm.get("Anitha"));

        System.out.println("Kavitha's value: " + hm.get("Kavitha"));

        System.out.println("Meera's value: " + hm.get("Meera"));

        System.out.println("Null key's value: " + hm.get(null));

        System.out.println("\nAll entries in HashMap:");

        for (Map.Entry<String, Integer> e : hm.entrySet())

            System.out.println(e.getKey() + " " + e.getValue());

        // Making HashMap synchronized

        Map<String, Integer> syncMap = Collections.synchronizedMap(hm);

        System.out.println("\nSynchronized Map entries:");

        for (Map.Entry<String, Integer> e : syncMap.entrySet())

            System.out.println(e.getKey() + " " + e.getValue());

    }

}

**Output:**