**Tasks:**

1. **Basic Types:**
   * Declare variables of the following types and assign appropriate values:
     + **number**
     + **string**
     + **boolean**
     + **array** of **numbers**
     + **tuple** with elements of type **string** and **number**
     + **enum** for days of the week

**CODE:**

**let mya: number = 10;**

**let myb: string = "Hello";**

**let myc: boolean = true;**

**let myd: number[] = [1, 2, 3, 4, 5];**

**let mye: [string, number] = ["Monday", 1];**

**enum DaysOfWeek {**

**Monday,**

**Tuesday,**

**Wednesday,**

**Thursday,**

**Friday,**

**Saturday,**

**Sunday**

**}**

**let today: DaysOfWeek = DaysOfWeek.Sunday;**

**console.log(mya);**

**console.log(myb);**

**console.log(myc);**

**console.log(myd);**

**console.log(mye);**

**console.log(today);**

**OUTPUT:**

**10**

**Hello**

**true**

**[ 1, 2, 3, 4, 5 ]**

**[ 'Monday', 1 ]**

1. **Functions:**
   * Write a function called **add** that takes two parameters of type **number** and returns their sum.
   * Write a function called **capitalize** that takes a parameter of type **string** and returns the capitalized version of it.

**CODE:**

**function add(a:number,b:number){**

**console.log(a+b);**

**}**

**add(10,20);**

**function capitalize(a:string){**

**console.log(a.toUpperCase());**

**}**

**capitalize(‘asrith’);**

**OUTPUT:**

**ASRITH**

1. **Interfaces:**
   * Define an interface called **Person** with the following properties:
     + **name** of type **string**
     + **age** of type **number**
     + **email** of type **string**
   * Declare a variable **user** of type **Person** and initialize it with sample data.

**CODE:**

**interface Person{**

**name:string;**

**age:number;**

**email:string;**

**}**

**let user: Person = {**

**name: "Asrith",**

**age: 21,**

**email: "asrithpotnuru@gmail.com"**

**};**

**console.log(user.name);**

**console.log(user.age);**

**console.log(user.email);**

**OUTPUT:**

**Asrith**

**21**

**asrithpotnuru@gmail.com**

1. **Classes:**
   * Create a class called **Car** with the following properties:
     + **make** of type **string**
     + **model** of type **string**
     + **year** of type **number**
   * Implement a method **displayInfo** which logs the information about the car.

**CODE:**

**class Car {**

**make: string;**

**model: string;**

**year: number;**

**constructor(make: string, model: string, year: number) {**

**this.make = make;**

**this.model = model;**

**this.year = year;**

**}**

**displayInfo(): void {**

**console.log("Car Information:\nMake: " + this.make + "\nModel: " + this.model + "\nYear: " + this.year);**

**}**

**}**

**let myCar: Car = new Car("Rolls Royce", "Q5", 2019);**

**myCar.displayInfo();**

**OUTPUT:**

**Car Information:**

**Make: Rolls Royce**

**Model: Q5**

**Year: 2019**

1. **Generics:**
   * Write a generic function called **reverseArray** that takes an array of any type and returns a reversed version of that array.

**CODE:**

**function reverseArray<As>(arr:As[]){**

**return arr.reverse();**

**}**

**let a: number[] = [1, 2, 3, 4, 5];**

**let b: number[] = reverseArray(a);**

**console.log(b);**

**OUTPUT:**

**[ 5, 4, 3, 2, 1 ]**

**TO GET OUPUT:**

**tsc filename.ts|node filename.ts**