1(a) TypeScript Program to Demonstrate Simple and Special Types

🎯Aim:

To write a TypeScript program that demonstrates the use of Simple (Primitive) and Special data types.

🧠 Theory:

TypeScript provides a rich set of data types to make programs more reliable and less error-prone.

Simple (Primitive) Types:

These represent basic values such as string, number, boolean, bigint, symbol, undefined, and null.

Special Types:

These provide flexibility and safety in dynamic programming situations. They include any, unknown, void, and never.

The any type allows any kind of data, while unknown requires checking before use. The void type is used for functions that return nothing, and never is used for functions that never return (like those throwing errors).

💻 Program:

```
// 1(a) TypeScript Program to Demonstrate Simple and Special Types
// ---------- SIMPLE TYPES ----------

// string
let studentName: string = "Sivateja";
console.log("String:", studentName);

// number
let studentAge: number = 21;
console.log("Number:", studentAge);

// boolean
let isEnrolled: boolean = true;
console.log("Boolean:", isEnrolled);
```

```
// bigint
let bigNumber: bigint = 1234567890123456789012345678901234567890n;
console.log("BigInt:", bigNumber);

// symbol
let uniqueId: symbol = Symbol("id");
console.log("Symbol:", uniqueId.toString());

// undefined
let notAssigned: undefined = undefined;
console.log("Undefined:", notAssigned);

// null
let emptyValue: null = null;
console.log("Null:", emptyValue);

// ---------- SPECIAL TYPES ----------

// any
let randomValue: any = "Hello TypeScript";
console.log("Any (string):", randomValue);
randomValue = 42;
console.log("Any (number):", randomValue);

// unknown
let unknownValue: unknown = "This is unknown";
if (typeof unknownValue === "string") {
  console.log("Unknown (after checking):", unknownValue.toUpperCase());
}

// void
```

```
function greet(): void {
  console.log("Void: This function returns nothing.");
}
greet();

// never
function throwError(message: string): never {
  throw new Error(message);
}
// Uncomment below line to test 'never' type
// throwError("This is a never type example");
```

🖥️ **Output:**

```
String: Sivateja
Number: 21
Boolean: true
BigInt: 123456789012345678901
Symbol: Symbol(id)
Undefined: undefined
Null: null
Any (string): Hello TypeScript
Any (number): 42
Unknown (after checking): THIS IS UNKNOWN
Void: This function returns nothing.
```

1(b) TypeScript Program Using Functions with Parameter and Return Type Annotations

Aim:
To write a TypeScript program that demonstrates the use of functions with parameter and return type annotations, including optional and default parameters.

Description:
This program defines functions in TypeScript using strict type annotations for parameters and return values. It includes an optional parameter (which can be skipped) and a default parameter (which assumes a predefined value if not provided).

Program:

```
// Functions with type annotations, optional, and default parameters
// Function with parameter and return type annotation
function add(a: number, b: number): number {
  return a + b;
}
// Function with an optional parameter
function greetUser(name: string, greeting?: string): string {
  return `${greeting || "Hello"}, ${name}!`;
}
// Function with default parameter
function multiply(a: number, b: number = 5): number {
  return a * b;
}
// Using the functions
console.log("Sum:", add(10, 20));
console.log(greetUser("Vivek"));
console.log(greetUser("Vivek", "Good Morning"));
console.log("Product:", multiply(4));
console.log("Product with both params:", multiply(4, 3));
```

Output:
(In Terminal)

Sum: 30

Hello, Vivek!

Good Morning, Vivek!

Product: 20

Product with both params: 12