**(Notes) Currying**

**Question 1: Currying**
**Explanation:** Currying is a technique in functional programming where a function with multiple arguments is transformed into a sequence of functions, each taking a single argument. This improves composability and allows partial application of functions.

---

**Question 2: sum(2)(6)(1)**
**Explanation:** This function demonstrates currying by taking three separate arguments and adding them together in a sequence using nested functions.

---

**Question 4: Write a currying function evaluate("sum")(4)(2)**
**Explanation:** This question showcases a curried function `evaluate` that takes an operation string and two numbers, then performs the corresponding operation (sum, multiply, divide, subtract) based on the string input.

---

**Question 5: Infinite Currying -> sum(1)(2)(3)....(n)**
**Explanation:** This example illustrates how currying can be used recursively to handle an indefinite number of arguments, continually adding them together until a termination condition is met.

---

**Question 6: Currying vs Partial Application**
**Explanation:** This question compares currying and partial application, showing how currying allows for creating reusable functions with specific arguments filled in, while partial application involves pre-filling some arguments and leaving others for later.

---

**Question 7: Real-world example of currying => Manipulating DOM**
**Explanation:** In this example, currying is used to create a function that updates the text content of an HTML element identified by its ID. This showcases a practical application of currying in web development for DOM manipulation.

---

**Question 8: Curry() implementation**
**Explanation:** This code snippet demonstrates a custom implementation of the `curry` function,

which transforms a multi-argument function into a curried function, enabling partial application and composability. It's applied to a simple sum function for demonstration.

---