```
<script>
// _valueA/B/C can be set to "SCAN", "RESPONSE", or
// "QUESTION"- compares first question asked regardless of ID
// "QUESTION_ID" i.e. "1234"- compares the specific question by ID
var _valueA = "SCAN";
var _valueB = "QUESTION";
var _valueC = false; // Leave false unless you want to do a three-way match

// Include __ORIGINAL__ in your override to show the original response text
var _validText = "MATCHED"; // Override is optional i.e. "Nice Job!\n__ORIGINAL__";
var _invalidText = "NOT MATCHED"; // Override is optional i.e. "Try Again";

// If you specify an error message and change var _invalidText to false, then a mismatch will
block submit.
var _errorMessage = false;

// When false, _valueA only needs to be a substring of _valueB
var isExactMatch = true;

// Appended prefix/suffix to _valueA before comparing
var prefixValueA = "";
var suffixValueA = "";

// Appended prefix/suffix to _valueB before comparing
var prefixValueB = "";
var suffixValueB = "";

// Appended prefix/suffix to _valueC before comparing
var prefixValueC = "";
var suffixValueC = "";

function getAnswerVal(qid, answerArray) {
    if (Array.isArray(answerArray)) {
        if (qid == false) {
            if (answerArray.length > 0 && Array.isArray(answerArray[0].value)) {
                return answerArray[0].value[0];
            }
            return answerArray[0]?.value || ""; // Prevents accessing undefined properties
        } else {
            var ans = answerArray.find(a => a.qid == qid);
            if (ans && Array.isArray(ans.value)) {
                return ans.value[0];
            }
            return ans?.value || ""; // Safe fallback
```

```
        }
    }
    return "";
}

function getValue(valType, dataJson) {
    valType = valType.toUpperCase();

    return valType == "SCAN" ? dataJson.scanValue :
           valType == "RESPONSE" ? dataJson.scanResponse :
           valType == "QUESTION" ? getAnswerVal(false, dataJson.answers) :
           getAnswerVal(valType, dataJson.answers);
    // If valType is other, then assume it's a question ID
}

function crCustomValidate(dataJson) {
    if (dataJson.scanStatus && dataJson.scanStatus == "1") { // Ensure scanStatus is defined
        var primary = prefixValueA + getValue(_valueA, dataJson) + suffixValueA;
        var secondary = prefixValueB + getValue(_valueB, dataJson) + suffixValueB;

        // Ensure secondary is a string before using indexOf()
        var isValid = isExactMatch
            ? primary == secondary
            : (typeof secondary == "string" && secondary.indexOf(primary) !== -1);

        if (isValid && _valueC !== false) {
            var tertiary = prefixValueC + getValue(_valueC, dataJson) + suffixValueC;

            // Ensure tertiary is a string before using indexOf()
            isValid = isExactMatch
                ? primary == tertiary
                : (typeof tertiary == "string" && tertiary.indexOf(primary) !== -1);
        }

        var obj = {
            "scanStatus": isValid ? "1" : "0",
            "scanResponse": dataJson.scanResponse || "" // Ensure scanResponse is not undefined
        };

        if (isValid && _validText !== false) {
            obj.scanResponse = _validText.replace("__ORIGINAL__", dataJson.scanResponse ||
"");
        } else if (!isValid) {
            if (_invalidText !== false) {
```

```
            obj.scanResponse = _invalidText.replace("__ORIGINAL__", dataJson.scanResponse
|| "");
        } else if (_errorMessage !== false) {
            obj.errorMessage = _errorMessage.replace("__ORIGINAL__",
dataJson.scanResponse || "");
        }
    }

    return JSON.stringify(obj);
  }
  return JSON.stringify(dataJson);
}
</script>
```