```
<script language = "JavaScript">

    // *** FUNCTIONS ***

    function MyDebuggingTool (MDT_Indicator) {

        // Debuging tool (for testing only!).

        // document.write ("My Debugging Tool: " + MDT_Indicator +"<br><br>");

    }

    function FormatDataError () {

        document.write ("Erreur dans le format des données!<br><br>");

    }

    function DumpDecisionTable (DDT, DDT_NumberOfRecords) {

        var i;

        document.write ("Dump of DecisionTable:<br><br>");

        document.write ("| --E-- | --0-- | --1-- |<br>| wdsss | wdsss | wdsss |<br><br>");

        // Note that record 0 is ignored ("i=1" instead of "i=0" in the "for loop" statement here below).

        for (i=1; i<DDT_NumberOfRecords; i++) {

        document.write (" | " +

            DDT[i][0][0] + DDT[i][0][1] + DDT[i][0][2] + " | " +
            DDT[i][1][0] + DDT[i][1][1] + DDT[i][1][2] + " | " +
            DDT[i][2][0] + DDT[i][2][1] + DDT[i][2][2] + " | = step #" + i + "<br>");

            }

        document.write ("<br>");
```

```
}

function InitDecisionTable (IDT, IDT_StepNumber,

    IDT_EmptyWriteData, IDT_EmptyDirectionData, IDT_EmptyStepData,
    IDT_ZeroWriteData,  IDT_ZeroDirectionData,  IDT_ZeroStepData,
    IDT_OneWriteData,   IDT_OneDirectionData,   IDT_OneStepData) {

    IDT[IDT_StepNumber][0][0] = IDT_EmptyWriteData;
    IDT[IDT_StepNumber][0][1] = IDT_EmptyDirectionData;
    IDT[IDT_StepNumber][0][2] = IDT_EmptyStepData;

    IDT[IDT_StepNumber][1][0] = IDT_ZeroWriteData;
    IDT[IDT_StepNumber][1][1] = IDT_ZeroDirectionData;
    IDT[IDT_StepNumber][1][2] = IDT_ZeroStepData;

    IDT[IDT_StepNumber][2][0] = IDT_OneWriteData;
    IDT[IDT_StepNumber][2][1] = IDT_OneDirectionData;
    IDT[IDT_StepNumber][2][2] = IDT_OneStepData;

}

function InitStripOfTape (ISOT, ISOT_BinaryValue) {

    var i;

    if (ISOT_BinaryValue.length !== 5) {FormatDataError ()};

    for (i = 0; i < 33; i++) {ISOT[i] = "_"};

    ISOT[18] = ISOT_BinaryValue.substring(0,1);
    ISOT[19] = ISOT_BinaryValue.substring(1,2);
    ISOT[20] = ISOT_BinaryValue.substring(2,3);
    ISOT[21] = ISOT_BinaryValue.substring(3,4);
    ISOT[22] = ISOT_BinaryValue.substring(4,5);

    }
```

```
function DumpStripOfTape (DSOT) {

    var i;

    document.write("--------------------------------O--------------------------------<br>");

    for (i = 0; i < 32; i++) {document.write(DSOT[i] + "|")}

    document.write(DSOT[32] + "<br>"); // pour éviter d'afficher le dernier bâton vertical ("|").

    document.write("--------------------------------O--------------------------------<br><br>");

    }

// ******************************************************
// ***                                                ***
// ***   THIS FUNCTION DECODES ANY TABLE OF DECISION   ***
// ***                                                ***
// ******************************************************

function ExecutingTableOfDecision (ETOD, ETOD_StepNumber, ETOD_StripOfTape) {

    switch (ETOD_StripOfTape[16]) {

        case "_":

            switch (ETOD[ETOD_StepNumber][0][0]) {

                // Write (nothing, 0 or 1 or _)

                case ".": MyDebuggingTool ("_ + nothing to write"); break;

                case "0": WriteZeroInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("_ + write 0"); break;

                case "1": WriteOneInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("_ + write 1"); break;

                case "_": EraseDataInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("_ + write _"); break;

            }
```

```
        switch (ETOD[ETOD_StepNumber][0][1]) {

            // Direction (nowhere, to the left, to the right)

            case ".": MyDebuggingTool ("_ + go to nowhere"); break;

            case "L": ShiftLeftStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("_ + go to the left"); break;

            case "R": ShiftRightStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("_ + go to the right"); break;

        }

        // Stop (999) or next step

        DecisionTableStepNumber = parseInt (ETOD[ETOD_StepNumber][0][2]);

    break;

    case "0":

        switch (ETOD[ETOD_StepNumber][1][0]) {

            // Write (nothing, 0 or 1 or _)

            case ".": MyDebuggingTool ("0 + nothing to write"); break;

            case "0": WriteZeroInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("0 + write 0"); break;

            case "1": WriteOneInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("0 + write 1"); break;

            case "_": EraseDataInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("0 + write _"); break;

        }

        switch (ETOD[ETOD_StepNumber][1][1]) {

            // Direction (nowhere, to the left, to the right)
```

```
            case ".": MyDebuggingTool ("0 + go to nowhere"); break;
            case "L": ShiftLeftStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("0 + go to the left"); break;
            case "R": ShiftRightStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("0 + go to the right"); break;
        }
        // Stop (999) or next step
        DecisionTableStepNumber = parseInt (ETOD[ETOD_StepNumber][1][2]);
    break;
    case "1":
        switch (ETOD[ETOD_StepNumber][2][0]) {
            // Write (nothing, 0 or 1)
            case ".": MyDebuggingTool ("1 + nothing to write"); break;
            case "0": WriteZeroInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("1 + write 0"); break;
            case "1": WriteOneInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("1 + write 1"); break;
            case "_": EraseDataInStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("1 + write _"); break;
        }
        switch (ETOD[ETOD_StepNumber][2][1]) {
            // Direction (nowhere, to the left, to the right)
            case ".": MyDebuggingTool ("1 + go to nowhere"); break;
            case "L": ShiftLeftStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("1 + go to the left"); break;
            case "R": ShiftRightStripOfTape (ETOD_StripOfTape); MyDebuggingTool ("1 + go to the right"); break;
```

```
            }

            // Stop (999) or next step

            DecisionTableStepNumber = parseInt (ETOD[ETOD_StepNumber][2][2]);

        break;

    }

}

function WriteZeroInStripOfTape (WZISOT) {WZISOT[16] = "0"; DumpStripOfTape (WZISOT);}

function WriteOneInStripOfTape (WOISOT) {WOISOT[16] = "1"; DumpStripOfTape (WOISOT);}

function EraseDataInStripOfTape (EDISOT) {EDISOT[16] = "_"; DumpStripOfTape (EDISOT);}

function ShiftLeftStripOfTape (SLSOT) {

    var i;

    for (i = 1; i < 32; i++) {SLSOT[i] = SLSOT[i+1]};

    DumpStripOfTape (SLSOT);

}

function ShiftRightStripOfTape (SRSOT) {

    var i;

    for (i = 1; i < 32; i++) {SRSOT[32-i] = SRSOT[32-(i+1)]};

    DumpStripOfTape (SRSOT);

}
```

```
// *** DECLARATIONS ***

var DecisionTable = new Array (128);

var StripOfTape = new Array(33);

var DecisionTableStepNumber;

var i;

for (i=0; i<128; i++) {DecisionTable[i] = new Array (3)};

for (i=0; i<128; i++) {

    DecisionTable[i][0] = new Array (3);
    DecisionTable[i][1] = new Array (3);
    DecisionTable[i][2] = new Array (3)};

for (i=0; i<128; i++) {

    DecisionTable[i][0][0] = ".";
    DecisionTable[i][0][1] = ".";
    DecisionTable[i][0][2] = "...";
    DecisionTable[i][1][0] = ".";
    DecisionTable[i][1][1] = ".";
    DecisionTable[i][1][2] = "...";
    DecisionTable[i][2][0] = ".";
    DecisionTable[i][2][1] = ".";
    DecisionTable[i][2][2] = "..."};

//DumpDecisionTable (DecisionTable, 8);

// *** INIT DECISION TABLE FOR "ADD 1" PROGRAM ***

/*

+-----------------------------------------------------+
|                      step #1                        |
+-----------------+-----------------+-----------------+
```

```
|      empty      |        0        |        1        |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  .  |  L  | 001 |  .  |  .  | 002 |  .  |  .  | 002 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

+-------------------------------------------------------+
|                       step #2                         |
+-----------------+-----------------+-----------------+
|      empty      |        0        |        1        |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  .  |  R  | 003 |  .  |  L  | 002 |  .  |  L  | 002 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

+-------------------------------------------------------+
|                       step #3                         |
+-----------------+-----------------+-----------------+
|      empty      |        0        |        1        |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  1  |  .  | 999 |  1  |  .  | 999 |  0  |  R  | 003 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

*/

InitDecisionTable (DecisionTable, 1, ".", "L", "001", ".", ".", "002", ".", ".", "002");
InitDecisionTable (DecisionTable, 2, ".", "R", "003", ".", "L", "002", ".", "L", "002");
InitDecisionTable (DecisionTable, 3, "1", ".", "999", "1", ".", "999", "0", "R", "003");

//DumpDecisionTable (DecisionTable, 8);

document.write ("<br><br><br>");
document.write ('<br><br><br><font color = "#ffe5bf">');
document.write ("-----------------------------------<br>");
document.write ("--- 1. Ajouter 1 à 111 (binaire) ---<br>");
document.write ("-----------------------------------<br><br></font>");

// *** INITIALIZING STRIP OF TAPE WITH BINARY NUMBER(S) ***

InitStripOfTape (StripOfTape, "__111");
```

```
    document.write ("<br>• Contenu initial du ruban:<br><br>");

    DumpStripOfTape (StripOfTape);

    DecisionTableStepNumber_Previous = 0;

    DecisionTableStepNumber = 1;

    while (DecisionTableStepNumber != 999) {

        if (DecisionTableStepNumber_Previous != DecisionTableStepNumber) {

            document.write ("<br>• Déroulement de l'étape numéro " + DecisionTableStepNumber + ":<br><br>");

            DecisionTableStepNumber_Previous = DecisionTableStepNumber;

        }

        ExecutingTableOfDecision (DecisionTable, DecisionTableStepNumber, StripOfTape);

    }

    document.write ("<br><br><br>");
    document.write ('<br><br><br><font color = "#ffe5bf">');
    document.write ("-------------------------------------<br>");
    document.write ("--- 2. Ajouter 1 à 110 (binaire) ---<br>");
    document.write ("-------------------------------------<br><br></font>");

    // *** INITIALIZING BINARY VALUE (TO SCALED ON 5 CHARACTERS!) ***

    InitStripOfTape (StripOfTape, "__110");

    document.write ("<br>• Contenu initial du ruban:<br><br>");

    DumpStripOfTape (StripOfTape);

    DecisionTableStepNumber_Previous = 0;

    DecisionTableStepNumber = 1;
```

```
while (DecisionTableStepNumber != 999) {

    if (DecisionTableStepNumber_Previous != DecisionTableStepNumber) {

        document.write ("<br>• Déroulement de l'étape numéro " + DecisionTableStepNumber + ":<br><br>");

        DecisionTableStepNumber_Previous = DecisionTableStepNumber;

    }

    ExecutingTableOfDecision (DecisionTable, DecisionTableStepNumber, StripOfTape);

}

document.write ("<br><br><br>");
document.write ('<br><br><br><font color = "#ffe5bf">');
document.write ("-------------------------------------<br>");
document.write ("--- 3. Ajouter 1 à 101 (binaire) ---<br>");
document.write ("-------------------------------------<br><br></font>");

// *** INITIALIZING BINARY VALUE (TO SCALED ON 5 CHARACTERS!) ***

InitStripOfTape (StripOfTape, "__110");

document.write ("<br>• Contenu initial du ruban:<br><br>");

DumpStripOfTape (StripOfTape);

DecisionTableStepNumber_Previous = 0;

DecisionTableStepNumber = 1;

while (DecisionTableStepNumber != 999) {

    if (DecisionTableStepNumber_Previous != DecisionTableStepNumber) {

        document.write ("<br>• Déroulement de l'étape numéro " + DecisionTableStepNumber + ":<br><br>");
```

```
        DecisionTableStepNumber_Previous = DecisionTableStepNumber;

    }

    ExecutingTableOfDecision (DecisionTable, DecisionTableStepNumber, StripOfTape);

}

document.write ("<br><br><br>");
document.write ('<br><br><br><font color = "#ffe5bf">');
document.write ("-----------------------------------<br>");
document.write ("--- 4. Ajouter 1 à 1 (binaire) ---<br>");
document.write ("-----------------------------------<br><br></font>");

// *** INITIALIZING BINARY VALUE (TO SCALED ON 5 CHARACTERS!) ***

InitStripOfTape (StripOfTape, "____1");

document.write ("<br>• Contenu initial du ruban:<br><br>");

DumpStripOfTape (StripOfTape);

DecisionTableStepNumber_Previous = 0;

DecisionTableStepNumber = 1;

while (DecisionTableStepNumber != 999) {

    if (DecisionTableStepNumber_Previous != DecisionTableStepNumber) {

        document.write ("<br>• Déroulement de l'étape numéro " + DecisionTableStepNumber + ":<br><br>");

        DecisionTableStepNumber_Previous = DecisionTableStepNumber;

    }

    ExecutingTableOfDecision (DecisionTable, DecisionTableStepNumber, StripOfTape);

}
```

```
    document.write ("<br><br><br>");
    document.write ('<br><br><br><font color = "#ffe5bf">');
    document.write ("----------------------------------<br>");
    document.write ("--- 5. Ajouter 1 à 0 (binaire) ---<br>");
    document.write ("----------------------------------<br><br></font>");

    // *** INITIALIZING BINARY VALUE (TO SCALED ON 5 CHARACTERS!) ***

    InitStripOfTape (StripOfTape, "____0");

    document.write ("<br>• Contenu initial du ruban:<br><br>");

    DumpStripOfTape (StripOfTape);

    DecisionTableStepNumber_Previous = 0;

    DecisionTableStepNumber = 1;

    while (DecisionTableStepNumber != 999) {

        if (DecisionTableStepNumber_Previous != DecisionTableStepNumber) {

            document.write ("<br>• Déroulement de l'étape numéro " + DecisionTableStepNumber + ":<br><br>");

            DecisionTableStepNumber_Previous = DecisionTableStepNumber;

        }

        ExecutingTableOfDecision (DecisionTable, DecisionTableStepNumber, StripOfTape);

    }

</script>
```