JavaScript　第3Q　単位認定試験（案）

# 次の文章について正しいものには〇、誤りには×を答えなさい。 （２０点　＠２点）

1. 配列の宣言を行う時には、Arrayキーワードを使用する。
2. 宣言された配列の初期値は、すべて０となる。
3. 2次元配列を作るときには必ず、for文を使って作らないといけない。
4. JavaScriptでは、N次元配列を作成することができる（Nは任意）。
5. 配列は４×３などのサイズが異なる配列は作成できない。
6. 要素数が４個の１次元配列aryは「var ary = new Array（４）」と書いて宣言を行う
7. 要素数が５×２の２次元配列aryは「var ary = new Array(5x2)」と書いて宣言を行う。
8. 予想が用意されていない配列にアクセスしたときには、その値は０となる。
9. ２次元配列の要素の数が多くなるときは、forもしくはwhile文を使用して、初期化を行うこともできる。
10. 配列の添え字は０から始まる

【解答欄】

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 番号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 解答 | 〇 | × | × | 〇 |  |  |  |  |  |  |

# 次の式の値について答えなさい。ただし、変数aは８、変数ｂは２３とする。 （２０点　＠２点）

1. a == b
2. a <= b
3. a <= ( b % 2 )
4. !( a != b )
5. ( a == 8 ) || ( b != 23 )
6. ( a != 8 ) && ( b == 23 )
7. a += b
8. ++ a = b ++

【解答欄】

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 番号 | 1 | 2 | 3 | 4 | 5 |
| 解答 | false | true | false | false | true |
| 番号 | 6 | 7 a | 7 b | 8 a | 8 b |
| 解答 |  |  |  |  |  |

# 問題に示される関数を作成しなさい。 （３２点　＠２点）

1. 関数名をInitを作成しなさい。

引数無し、変数pos\_racket\_xを変数screen\_wの半分にする。

変数pos\_Racket\_y に screenH から 変数WALL\_THICKNESS \* 4を引いた値をセットする  
関数drawBall()を呼び出す

関数drawRacket()を呼び出す

1. 関数Init2を作成しなさい

関数document.getElementById()を使用し、HTMLの中から"myCanvas"要素を取り出し、変数itemに格納する

変数itemの要素が示すwidth属性の値を関数getAttributte()で取り出し、文字"px"を関数replace()を使用して取り除き、変数screen\_wに格納する。1倍して数値化すること。

変数itemの要素が示すheight属性の値を取り出し、文字"px"を取り除き、変数screen\_hに格納する1倍して数値化すること。

1. 関数MakeMaze（）を作成しなさい。  
   引数は使用しない。  
   ２次元配列の変数として、変数mazeを使用する

縦１０１、横１０１の迷路を作るとき、２次元配列を使用して、迷路を作成する。２次元配列の名前をmazeとするとき、２次元配列を作成し、０で初期化しなさい。  
迷路のサイズ１０１は変数MazeSizeに格納されており、プログラムの中で101は使用しないこと。

【解答欄】

|  |  |
| --- | --- |
| 問題 | プログラム |
| (1) | function Init(){ |
| pos\_racket\_x = screen\_w　/ 2 |
| pos\_racket\_y = WALL\_THICKNESS \* 4 |
| drawBall() |
| drawRacket() |
| } |
| (2) | function Init2(){ |
| var item = document.getElementById( "myCanvas") |
| screen\_w = item.getAttribute( "width").replace( "px", "") \* 1 |
| screen\_h = item.getAttribute( "height").replace( "px", "") \* 1 |
| } |

|  |  |
| --- | --- |
| (3) | function MakeMaze(){ |
| maze = new Array( MazeSize ) |
| for ( var i = 0 ; i < MazeSize ; i++ ){ |
| maze[ i ] = new Array( MazeSize ) |
| } |
| for ( var i = 0 ; i < MazeSize ; i++ ){ |
| for ( var j = 0 ; j < MazeSize ; j++){ |
| maze[i][j] = 0 |
| } |
| } |
| } |

# 次のプログラムを読み、質問に答えなさい。 （２８点　）

|  |  |
| --- | --- |
| 行番号 | コード |
| 1 | // make 柱 |
| 2 | for (var i = 2; i < MAZE\_AREA\_SIZE; i += 2) { |
| 3 | for (var j = 2; j < MAZE\_AREA\_SIZE; j += 2) { |
| 4 | ary[i][j] = CONST\_WALL |
| 5 | } |
| 6 | } |
| 7 | // make wall |
| 8 | for (var i = 2; i < MAZE\_AREA\_SIZE - 1; i += 2) { |
| 9 | for (var j = 2; j < MAZE\_AREA\_SIZE - 1; j += 2) { |
| 10 | var houkou = makeRandom4() |
| 11 | switch (houkou) { |
| 12 | case 1: |
| 13 | ary[i - 1][j] = CONST\_WALL |
| 14 | break |
| 15 | case 2: |
| 16 | ary[i + 1][j] = CONST\_WALL |
| 17 | break |
| 18 | case 3: |
| 19 | ary[i][j - 1] = CONST\_WALL |
| 20 | break |
| 21 | case 4: |
| 22 | ary[i][j + 1] = CONST\_WALL |
| 23 | break |
| 24 | } |
| 25 | } |
| 26 | } |

1. このプログラムを動作させるためには、関数makeRandom4()を作成する必要がある。この関数によって計算される値の最小値と最大値について答えなさい。
2. このプログラムにおいて(i,j)の組み合わせが、（6,4）で、makeRandom4()の計算結果が３のとき、壁はどのように作られるか答えなさい。他の壁からの影響は無視してよい。
3. このプログラムを使用して迷路を作成するとき、簡単なアルゴリズムで迷路を成績できる一方で、迷路の生成に無駄な点が生じる。この迷路生成における無駄な点について説明しなさい。

【解答欄】

|  |  |  |
| --- | --- | --- |
| 問題番号 | |  |
| 1 | 最大値 | １ |
| 最小値 |  |
| 2 | | セルの値の配置   |  |  |  | | --- | --- | --- | | （5,3） | (6,3) | (7,3) | | (5,4) | (6,4)  CONST\_WALL | (7,4) | | (5,5) | (6,5) | (7,5) |   　※CONST\_WALLは１と同じ |
| 3 | |  |