Task: Spreadsheet Evaluator

Implement an application that is able to evaluate data structure representing a spreadsheet.

Hub API

Your application has to connect to the hub, download a batch of jobs, compute them and submit them back. See hub URL below, all paths are relative to it. Note that some URLs are dynamic and can only be retrieved through the API.

Hub URL: https://www.wix.com/_serverless/hiring-task-spreadsheet-evaluator

`GET /jobs`	
	`// No request body.`
`HTTP 200`	

`{`
`  "submissionUrl": "...",`
`  "jobs": [`
`  { "id": "j1", data: ... },`
`  { "id": "j2", data: ... }`
`  ]`
`}`
	`// Job ID=j1`
	`// Job ID=j2`

`POST <submissionUrl>`	

`{`
`  "email": "pijusn@wix.com",`
`  "results": [`
`  { "id": "j1", data: ... },`
`  { "id": "j2", data: ... }`
`  ]`
`}`
	`// Use your email here.`
	
	`// Result of job ID=j1`
	`// Result of job ID=j2`

`HTTP 200`	

`{`
`  "message": "Good job!"`
`}`	

Notes:

- if your submission is incorrect, you will get an error message instead.

- you can submit multiple times - no associated penalty.

Spreadsheet Data Structure (aka "The Job")

Application's purpose is to evaluate a spreadsheet and return a fully evaluated spreadsheet back. A spreadsheet is represented as a 2-dimensional array of cells. Cells can be referred to using A1 notation.

Spreadsheet	Comments

`[`
`  [`
`  { … },`
`  { … },`
`  ],`
`  [`
`  { … },`
`  { … },`
`  ],`
`]`
	
	`Cell A1`
	`Cell B1`
	
	`Cell A2`
	`Cell B2`

Value Cells

Value cells represent constant values. They can be one of 3 subtypes defined in the table.

Subtype	Example
Number value	`{ "value": { "number": 45.8 } }`
Text value	`{ "value": { "text": "hire me!" } }`
Boolean value	`{ "value": { "boolean": true } }`

Typing is strict and no implicit conversions between the types is possible. This is mostly to simplify the implementation.

Error Cells

Error cells represent an error that occurred during evaluation. In essence, it's just an error message describing what went wrong.

Type	Example
Error	`{ "error": "invalid reference" }`

Formula Cells

These are the complicated ones. Formula is a composite data structure that can (in most cases) be evaluated to a constant value. It means that after evaluation, the formula cell is replaced by either a value cell or error cell. Formula structure is very strict (to simplify the implementation). In case input structure is invalid, types do not match or refer to non-existing cells - result is an error. Error messages are not defined in this document and therefore not checked by the hub.

Formula cells will always have the following structure. Property value will be one of the operators defined below.

Type	Examples
Formula	`{ "formula": { "value": { "number": 42 } } }`
	`{ "formula": { "reference": "A2" } }`
	`{ "formula": { "sum": [ { "value": { "number": 2 } }, { "reference": "B1" } ] } }`

Leaf Nodes

Operator	Comment	Example
value	Evaluates to a constant value. Structure is of a value cell.	`{ "value": { "number": 42 } }`
reference	Resolves to a cell value referred to using A1 notation.	`{ "reference": "C5" }`

Tree Nodes (operators)

Operator	Comment	Example
`sum`	Add one or more values together.	`{ "sum": [ { "value": { "number": 15 } }, { "reference": "A1" } ] }`
`multiply`	Multiplies one or more numbers.	`{ "multiply": [ { "value": { "number": -1 } },`

		<pre>{ "reference": "A1" }
		]
		}</pre>

divide	Mathematical division. Divides two values (first divided by second).
	Acceptable error: 10^{-7}
		<pre>{
		"divide": [
		{ "value": { "number": 6 } },
		{ "value": { "number": 4 } },
		]
		}</pre>

is_greater	Returns true if the first operand is greater than the second operand.	<pre>{
		"is_greater": [
		{ "value": { "number": 6 } },
		{ "value": { "number": 4 } },
		]
		}</pre>

is_equal	Returns true if the first operand is equal to the second operand.	<pre>{
		"is_equal": [
		{ "value": { "number": 6 } },
		{ "value": { "number": 4 } },
		]
		}</pre>

not	Negates a boolean value.	<pre>{
		"not": { "value": { "boolean": true } }
		}</pre>

and	Logical *and* operation. True if all parameters are true.	<pre>{
		"and": [
		{ "reference": "A1" },
		{ "reference": "B1" }
		]
		}</pre>

or	Logical *or* operation. True if at least one parameter is true.	<pre>{
		"or": [
		{ "reference": "A1" },
		{ "reference": "B1" }
		]
		}</pre>

if	Conditional. Arguments:
	1 - condition. Must be a boolean.
	2 - value if condition is true.
	3 - value if condition is false.
		<pre>{
		"if": [
		{
		"is_greater": [
		{ "reference": "A1" },
		{ "reference": "B1" },
		]
		},
		{ "value": { "text": "Stonks!" } },
		{ "value": { "text": ":(" } }
		]
		}</pre>

concat	Concatenates strings together.	<pre>{
		"concat": [
		{ "value": { "text": "Hello" } },
		{ "value": { "text": ", " } },
		{ "value": { "text": "World!" } },
		]
		}</pre>

A1 Notation

This is a more strict (than usual) variation of A1 notation in order to simplify the implementation. This notation allows referencing a single cell in the spreadsheet. It will always match a regular expression `[A-Z][0-9]+`.

Letter represents column - numbered A to Z. Number afterwards represents row number, numbered 1 and upwards.

Examples

Simple

Jobs Received

```
{
  "submissionUrl": "https://wix.com/_something/submit",
  "jobs": [
    {
      "id": "j123",
      "data": [
        [
          { "value": { "number": 6 } },
          { "value": { "number": 4 } },
          {
            "formula": {
              "sum": [
                { "reference": "A1" },
                { "reference": "B1" }
              ]
            }
          }
        ]
      ]
    }
  ]
}
```

```
// Cell A1 ( =6 )
// Cell B1 ( =4 )
// Cell C1 ( =A1 + B1 )
```

Results submitted

```
{
  "email": "pijusn@wix.com",
  "results": [
    {
      "id": "j123",
      "data": [
        [
          { "value": { "number": 6 } },
          { "value": { "number": 4 } },
          { "value": { "number": 10 } },
        ]
      ]
    }
  ]
}
```

```
// Cell A1 ( =6 )
// Cell B1 ( =4 )
// Cell C1 ( =10 )
```

`Mismatching types`

Jobs Received

```
{
  "submissionUrl": "https://wix.com/_something/submit",
  "jobs": [
    {
      "id": "j123",
      "data": [
        [
          { "value": { "boolean": true } },
          { "value": { "number": 1 } },
          {
            "formula": {
              "and": [
                { "reference": "A1" },
                { "reference": "B1" }
              ]
            }
          }
        ]
      ]
    }
  ]
}
```

```
// Cell A1 ( =true )
// Cell B1 ( =1 )
// Cell C1 ( =AND(A1, B1) )
```

Results submitted

```
{
  "email": "pijusn@wix.com",
  "results": [
    {
      "id": "j123",
      "data": [
        [
          { "value": { "boolean": true } },
          { "value": { "number": 1 } },
          { "error": "type does not match" },
        ]
      ]
    }
  ]
}
```

```
// Cell A1 ( =true )
// Cell B1 ( =1 )
// Cell C1 ( ERROR )
```