


# CISC 131
# Introduction to Programming and Problem Solving
# Spring 2020
# Data Fields

**Due: Wednesday, April 29, 2020, at start of class**
**Points: none**

## Assignment

In part of this assignment you will write a function that is passed a *String* and returns an array. You have done something like this before in a function that returned an array each element of which contained one character of the *String*. The function you will write for this assignment is quite different than the one you wrote earlier.

A *String* can be formatted to contain fields. A *field* is a piece of information such as a name or age or salary. Here's an example:

```
Bob;Smith;21
```

The *String* contains three fields: first name, last name, and age. Other than the last field in the *String*, the end of each field is marked with a *delimiter*.

A *delimiter* is a character (or, sometimes, a character sequence) that identifies the end of the information for a field. It is not part of the information. It appears immediately following the information.

In the above example, the delimiter is the *semi-colon*. The last field of the *String* is never followed by a delimiter. The end of the String implicitly delimits the last field in the String. To find out how many fields are in a *String*, you count the number of delimiters in the *String* and then add one to that count.

The delimiter can be any character and someone needs to tell you what it is or you won't be able to extract the fields. How many fields are in this *String*?

```
Abc.21.5,0,later.time
```

Since any character or characters can be a delimiter, it is really impossible to tell. If the period is the delimiter, there are four fields; if the comma is the delimiter, there are three fields; if both the comma and the period are delimiters, then there are six fields. The delimiter need not be a punctuation character. If the letter *a*[1] were the delimiter, the above *String* would have two fields.

If the *String* contains consecutive delimiters like this example in which the comma is used as the delimiter:

```
Square,green,,3
```

there are four fields. The third field has no characters in it. The comma is often used as a delimiter. *Comma separated values* (CSV) is a common format used by *Excel* and other software to exchange information in text form.

The first function you will write has this header:

[1] Remember, each letter has an unique UNICODE code value. *A* is not equal to *a*.

```
function getDelimiterLocations ( line, delimiter )
```

The *line* parameter is a *String* that could be *null* or could contain zero or more characters. The *delimiter* parameter is a *String* that always will contain one or more characters.

The function returns an array of integers. The number of elements in the array is the same as the number of fields in the *line* parameter. Each element in the array is the location (character number) of a delimiter in the *line* parameter: element zero of the array is the location of the delimiter of the first field in *line*, element one of the array is the location of the delimiter of the second field in *line*, *et cetera*. If the *line* parameter is *null* or contains no characters, the function returns a zero length array.

Since the array must contain the delimiter location of each field in *line* but the last field never contains a delimiter, what should you place in the last element of the array? Make sure you test using a *delimiter* parameter that contains more than one character. Test thoroughly. The other functions you will write depend on this one being correct. Here are some examples in which the comma is used as the delimiter:

| **character numbers in *line*** | | | | | | | | | | **value of elements in returned array** |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| | a | b | , | | e | , | w | f | | 3 6 9 |
| h | i | , | t | h | e | r | e | | | 2 8 |
| , | , | , | , | | | | | | | 0 1 2 3 4 |
| *null* or zero length | | | | | | | | | | |
| s | u | m | m | e | r | | | | | 6 |

Here is a funciton to display arrays. This was originally given in Lecture 8. It will help you find out what your *getDelimiterLocations* function is doing.

```
function arrayDisplay ( someArray )
{
 var i;
 var result;

 result = "Array contains " + someArray.length + " elements.";
 i = 0;
 while ( i < someArray.length )
 {
  result = result + "\n[" + i + "]: " + someArray[i];
  i      = i + 1;
 }

 window.alert ( result );
}  // arrayDisplay
```

The second function you will write has this header:

```
function split ( line, delimiter )
```

Again, the *line* parameter is a *String* that is either *null* or contains zero or more characters. The *delimiter* parameter is a *String* that contains one or more characters. The function returns an array of *Strings*. Element zero of the array is the first field in *line*; element one of the array is the second field in *line*, *et cetera*. While there is a *split* method that is part of the Javascript *String* class, you must **not** use it.

Test everything thoroughly. Display the contains of the arrays as you test so you can see what is happening.

**What You Must Hand In**

Nothing