Computer Science Correspondence School

discere domi

Lecture
14

CISC 131
Introduction to Programming and Problem Solving
Spring 2020
More Date Functions

Due: **Friday, April 24, 2020, at start of class**
Points: **none yet but will be part of an assignment soon**

Today's Assignment

Write and test the following functions but do not add them to the *Dates.js* library yet. You may not use any *String* processing in the implementations of these functions. Only the *getDaysInMonth* function creates an array. Make sure your *Dates.js* library is loaded with your html file because some of the functions you are about to write will need to use them.

```
function getDaysInMonth ( month, year )
```

This function is passed an integer month number (one through twelve) and an integer year number. It returns the number of days in that month. For example, if 3 and 1834 were passed to it when the function was called, it would return 31 because there were thirty-one days in March or 1834. The function must be written using an array to hold the days in each month. Be sure your implementation returns twenty-nine for February in leap years.

```
function getDayNumberInYear ( yyyymmdd )
```

This function is passed a date encoded in an integer. The function returns the day number in the year for that date. For example, January 1 in any year is day number 1 and December 31 is day 365 in non-leap years and 366 in leap years.. Your implementation may not create any arrays and must not use a *while* loop. Use a *for* loop. The function must make appropriate use of the *getDaysInMonth* function.

```
function tomorrow ( yyyymmdd )
```

This function is passed a date encoded in an integer. The function returns the date that corresponds to the day *following* the parameter date. The return value is an integer in the *yyyymmdd* format. For example, if the parameter date was 19981231 the function would return 19990101.

Test this one thoroughly be putting it in a loop. If something is incorrect, check to make sure your leap year calculation is correct and that you *getDaysInMonth* calculation is correct.

```
function yesterday ( yyyymmdd )
```

This function is passed a date encoded in an integer. The function returns the date that corresponds to the day *before* the parameter date. The return value is an integer in the *yyyymmdd* format. For example, if the parameter date was 19990101 the function would return 19981231.

This function is not as easy as the *tomorrow* function. Here is some test code:

```
var x;
var date;
var j
```

```
j    = 1000000;
x    = 12000101;
date = x;
for(i=0;i<j;i=i+1) date = tomorrow(date);
for(i=0;i<j;i=i+1) date = yesterday(date);
window.alert(x+"  "+date);
```

The test code starts on January 1, 1200 and calls the *tomorrow* function a million times. It then calls the *yesterday* a million times. At the end of the second loop it displays the starting date of January 1, 1200 and the end date after doing all those *yesterdays*. The two dates that are displayed should be the same. If they are not, change the value of *j* to something much smaller and think about what happens when you move back one day across a month boundary or across a year boundary. It is not as straightforward as the *tomorrow* function.