Complete a Program Development Worksheet (Setup, Coding & Validation) for each of the following:

1. **Vectorized Leap Day Function:** Submit the vectored Leap Day function from Section II.

Problem ID \_Vectorized Leap Day Programmer Solution

Set Up/ Introduction Type of Program:  Script ✓ Function

1. Problem Statement:

Make a function that can take a vector years, determine if they are leap years or not and return a vector of 0s and 1s. With ones representing leap years and zeros not leap years. (updating previous function to handle vectors element-by-element).

Inputs: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Units or Values | Input Source\* |
| year | A vector of years | Years | Command line |

1. Output: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Units or Values | Output type\* |
| day | A vector with ones for leap years | 0 or 1 | Command Line |

1. Solution Steps (order of these two parts may be varied):

|  |  |  |
| --- | --- | --- |
| **year (input)** |  | **day (output)** |
| 2000 | Divisible by 400 | 1 |
| 1900 | Divisible by 100  but not by 400 | 0 |
| 2008 | Divisible by 4 but not 100 | 1 |
| 2009 | Not divisible by 4 | 0 |

Perform calculation on test case(s) Identify the steps/equations

A reasonable attempt at describing steps or doing a flowchart should be accepted for full credit

1. Programming

function day=LeapDayV(year)

% This function will take any calendar year (full digits) and return a 1 if

% it is a leap year and a 0 if it is not a leap year. The function follows

% the Gregorian Calendar rules. The function can handle both single years

% and vectors of years.

%

% function day=LeapDayV(year)

% Input: year = a Gregorian Calendar year (all digits)

% Output: day = 1 for leap years, 0 for non leap years

% preallocate the length of the output vector

day = zeros([1,length(year)]);

% for looop to step through each member of the loop

for n = 1:length(year)

if mod(year(n), 400) == 0 % rule 1 leap year if divisible by 400

day(n) = 1;

elseif mod(year(n), 100) ==0 % rule 2 not leap if divisible by 100

day(n) =0;

elseif mod(year(n),4) ==0 % rule 3 leap if divisible by 4

day(n) = 1;

else

day(n) =0;

end

end

1. Validation

> day = LeapDayV([2000 1900 2008 2009])

day =

1 0 1 0

**Matches test case exactly**

1. **Vectorized Make-Even:** Convert the Make Even function from two weeks ago to handle vectors (i.e. make it so a vector of values can be input and the program will return a vector of even numbers).

Problem ID \_Vectorized Make Even Function Programmer \_S. Moor \_\_\_\_

Set Up/ Planning Type of Program:  Script 🗹 Function

1. Problem Statement:

Develop a function that will have an output of an even integer. If an integer is even then it is simply output as is. If an odd integer is input the program will add one and return the resulting integer. 

1. Inputs: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Units or Values | Input Source\* |
| x | Input Value | Must be Integer | Command Line |

1. Output: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Units or Values | Output type\* |
| x | Even output value | Integer | Command Line |

1. Solution Steps (order of these two parts may be varied):
2. Perform calculation on test case(s) (2) Identify the steps/equations to be used in code

A reasonable attempt at describing steps or doing a flowchart should be accepted for full credit.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | 2 |
| 3 | 4 |
| -4 | -4 |
| -3 | -2 |

1. Programming  
   function x = MakeEvenV(x)

% This function forces an integer input to be even by increasing any odd

% number by one. This function can handle vector input and make each

% element even.

% S. Scott Moor March 2020

%

% function x = MakeEvenV(x)

% input: x = any integer or integer vector

% output: x = an even integer or integer vector (same as input or one higher)

% loop steps through the vector one element at a time. Stores answer back

% in the elements position

for k = 1:length(x)

% Conditional tests if the input is even and adds one if it is not

if rem(x(k),2)~= 0

x(k) = x(k) + 1;

end

end

1. Validation   
   >> MakeEvenV([2 3 -4 -3])

ans = Matches hand result in # 4 exactly

2 4 -4 -2

1. **Arbitrary Vector** Write a program that takes an arbitrary vector of real numbers and returns a vector with negative numbers squared and positive numbers halved.

Problem ID \_Arbitrary Vector Programmer Solution

Set Up/ Introduction Type of Program:  Script ✓ Function

1. Problem Statement:

Develop a function that will take a vector of real numbers, square the negative numbers and divide the positive numbers by two. The program then returns a vector of the modified numbers.

1. Inputs: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Units or Values | Input Source\* |
| A | Input vector | Vector or real numbers | Command line |

1. Output: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Units or Values | Output type\* |
| r | Output vector with corresponding results | Vector of real numbers | Command line |

1. Solution Steps (order of these two parts may be varied):

Perform calculation on test case(s) Identify the steps/equations

Or use a flowchart for steps

Test vector: a = [4, -3, -2, 12, 1]

Expected result b= [2, 9, 4, 6, ½]

List of steps

1. Start with index = 1
2. Select element
3. Test if it is greater than zero
   1. If true 🡺 divide by two
   2. If false 🡺 square
4. Store result in equivalent place   
   in output vector.
5. If element number is < length of input  
   increment index and repeat steps 2 – 5

Or else if false end.

1. Programming:

function y = Arb(x)

% function y=Arb(x)

%

% For a positive number this function returns half of the number

% For a negative number it returns the square of the number.

% The function will handle vectors element-by-element

%

% S. Scott Moor October 2018

%

% Variables

% Input: x = an arbitrary vector of real numbers

% Output: y = the resulting transformation

% pre-allocation (optional)

y = zeros(size(x));

% for loop steps through the input vector & tests elements one-by-one

% Divides positive numbers by 2 and squares negative numbers.

for k = 1:length(x)

if x(k) >=0

y(k) = x(k)/2;

else

y(k) = x(k)^2;

end

end

1. Validation

>> a = [4, -3, -2, 12, 1]

a =

4 -3 -2 12 1

>> Arb(a)

ans =

2.0000 9.0000 4.0000 6.0000 0.5000

Result matches test expectations from step 4.

**Extra Credit: Fibonacci Series:** Prepare a program that will return the terms of a Fibonacci Sequence given the first two values and the number of terms. In a Fibonacci Series each successive term is the sum of the two terms that come before it. For example the 6 term series with initial terms of 1 and 1 is:

1, 1, 2, 3, 5, 8

Problem ID \_Fibonacci Series & the golden ratio Programmer Solution

Consulted with \_\_Problem adapted from Holly Moore, *MATLAB for Engineers*, (2007)

Set Up/ Introduction Type of Program:  Script ✓ Function

1. Problem Statement:

Develop a function that will return a Fibonacci Sequence given the first two numbers in the series and the total number of terms.

1. Inputs: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Units or Values | Input Source\* |
| n1 | First number in series | Integer | Command line |
| n2 | Second number in series | Integer | Command line |
| N | Number of terms in series | Integer | Command line |

1. Output: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Units or Values | Output type\* |
| FS | Fibonacci Series | integer series | Command line |

1. Solution Steps (order of these two parts may be varied):

Perform calculation on test case(s) Identify the steps/equations

For n1 = 1, n2 = 2 and N = 6

1. Set first two numbers into the series
2. Calculate the next term in the series from the previous term.
3. Repeat step 2 until the number of terms = N
4. Plot resulting ratios in order

Or use a flowchart

The series would be:

FS = 1, 1, 2, 3, 5, 8, 13, 21

5. Programming:

function FS = Fib(n1, n2, N)

% function FS = FibR(n1, n2, N)

% This function will create a Fibonacci Series, Fibonacci

% Sequences are series where each element is the sum of the previous two

% elements.

% S. Scott Moor October 2018

%

% Input Variables: n1 = the first number in the series (integer)

% n2 = the second number in the series (integer)

% N = number of terms in the series (integer)

% Output Variable: FS = Resulting Fibonacci Series

% Other Variables k = loop index and current term in series

% Preallocate the Fibonacci Series (optional)

FS = zeros(1,N);

% Place the first two terms in the series

FS([1,2]) = [n1, n2];

% Each iteration of this loop creates the next term in the series for

% terms 3 through N

for k = 3:N

FS(k) = FS(k-1)+FS(k-2);

end

6. Validation: Test Case Check:

>> FS = Fib(1,1,8)

FS =

1 1 2 3 5 8 13 21

These results match the test calculation with in significant figures use.