```
% ENGR 451 - Chapter 1 Laboratory
% Matlab tutorial

clear
x.data = [1 2 3 4 5];
x.offset = 1;
y.data = [5 4 3];
y.offset = 0;

% test add
test_lab1_2024('addit(x, y)')
test_lab1_2024('addit(y, x)')
test_lab1_2024('addit(1, x)')
test_lab1_2024('addit(x, 1)')

y.offset = 2;
test_lab1_2024('addit(x, y)')
test_lab1_2024('addit(y, x)')

y.offset = 5;
% test sub
test_lab1_2024('subit(x, y)')
test_lab1_2024('subit(y, x)')
test_lab1_2024('subit(1, x)')
test_lab1_2024('subit(x, 1)')

% test mult
test_lab1_2024('multit(x, y)')
test_lab1_2024('multit(3, x)')
test_lab1_2024('multit(x, 3)')

% test flip
test_lab1_2024('flipit(x)')

% test shift
test_lab1_2024('shiftit(y, 2)')

% test trim
x.data = [0 0 1 2 3 0];
test_lab1_2024('trimit(x)')

%combinations
test_lab1_2024('flipit(subit(shiftit(addit(x, 2), 4), y))')
test_lab1_2024('addit(flipit(addit(x, y)), shiftit(y, -5))')
test_lab1_2024('subit(addit(multit(shiftit(flipit(x), 4), shiftit(y, 3)),
flipit(y)), x)')

% test stem
clf
stemit(y)
grid on
ax = axis;
```

```
set(gca, 'XTick', ax(1):ax(2), ...
    'YTick', ax(3):ax(4))
ch = get(gca, 'Child');
ch.MarkerFaceColor = 'b';

% Program Listings
fprintf('\n\n')
disp('--- flipit.m --------------------------------')
type flipit
disp(' ')
disp('--- shiftit.m -------------------------------')
type shiftit
disp(' ')
disp('--- addit.m ---------------------------------')
type addit
disp(' ')
disp('--- subit.m ---------------------------------')
type subit
disp(' ')
disp('--- multit.m --------------------------------')
type multit
disp(' ')
disp('--- trimit.m --------------------------------')
type trimit
disp(' ')
disp('--- stemit.m --------------------------------')
type stemit
```

```
addit(x, y): sequence O.K.
addit(y, x): sequence O.K.
addit(1, x): sequence O.K.
addit(x, 1): sequence O.K.
addit(x, y): sequence O.K.
addit(y, x): sequence O.K.
subit(x, y): sequence O.K.
subit(y, x): sequence O.K.

z =

  struct with fields:

    data: [0 -1 -2 -3 -4]

subit(1, x): sequence O.K.
subit(x, 1): sequence O.K.
multit(x, y): sequence O.K.
multit(3, x): sequence O.K.
multit(x, 3): sequence O.K.
flipit(x): sequence O.K.
shiftit(y, 2): sequence O.K.
trimit(x): sequence O.K.
flipit(subit(shiftit(addit(x, 2), 4), y)): sequence O.K.
addit(flipit(addit(x, y)), shiftit(y, -5)): sequence O.K.
subit(addit(multit(shiftit(flipit(x), 4), shiftit(y, 3)), flipit(y)), x):
```

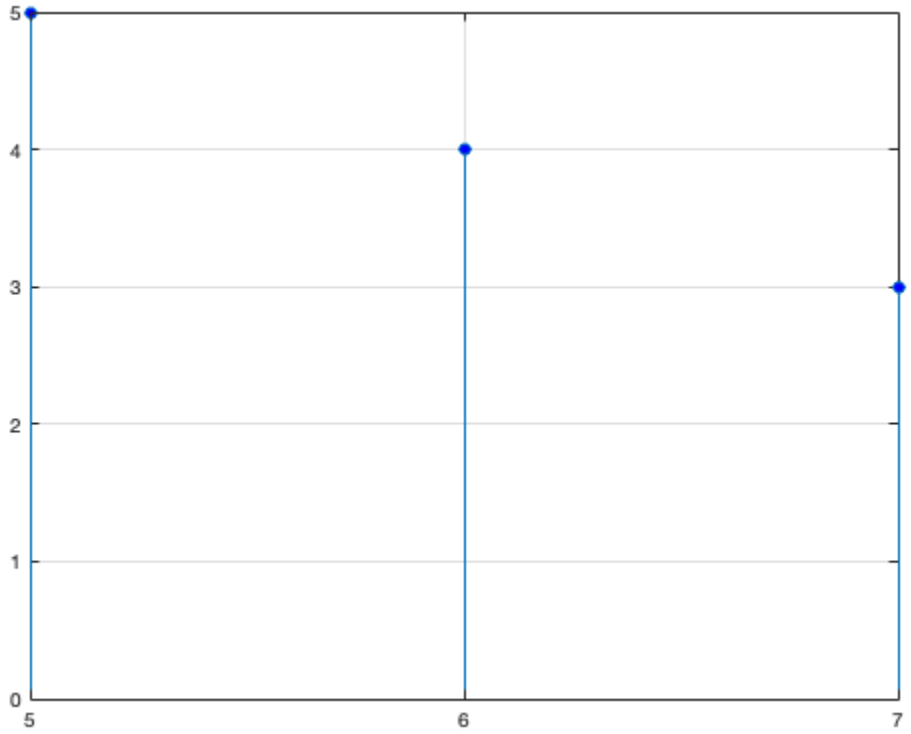
```
sequence O.K.


--- flipit.m --------------------------------

function y = flipit(x)
% FLIPIT Flip a Matlab sequence structure x so y = x[-n]
     y.data = flip(x.data);
     y.offset = -(length(x.data) + x.offset - 1);
  %  y = x(end:-1:1);
end

--- shiftit.m -------------------------------

function y = shiftit(x, n0)
% SHIFTIT Shift a Matlab sequence structure x by integer amount n0 so that
y[n] = x[n - n0]
    y.data = x.data;
    y.offset = x.offset + n0;

--- addit.m --------------------------------

function z = addit(x, y)
% ADDIT Add x and y. Either x and y will both be sequence structures or one
of them may be a number.
    % case where x or y is a number
    if isnumeric(x)
        z.data = x + y.data;
        z.offset = y.offset;
        z = trimit(z);
        return
    elseif isnumeric(y)
        z.data = x.data + y;
        z.offset = x.offset;
        z = trimit(z);
        return
    end

    % case where x and y are both sequence structures
    % make sure x and y are the same length
    if x.offset > y.offset
        % pad the front of x with zeros
        x.data = [zeros(1, x.offset - y.offset), x.data];

        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end

    elseif x.offset < y.offset
        % pad the front of y with zeros
```

```
        y.data = [zeros(1, y.offset - x.offset), y.data];

        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end
    else
        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end
    end

    % add the data
    z.data = x.data + y.data;
    % set the offset to the minimum of the two
    z.offset = min(x.offset, y.offset);
    z = trimit(z);

--- subit.m --------------------------------

function z = subit(x, y)
% SUBIT Subtract x and y.  Either x and y will both be sequence structures or
one of them may be a number.
    % case where x or y is a number
    if isnumeric(x)
        z.data = x - y.data
        z.offset = y.offset;
        z = trimit(z);
        return
    elseif isnumeric(y)
        z.data = x.data - y;
        z.offset = x.offset;
        z = trimit(z);
        return
    end

    % case where x and y are both sequence structures
    % make sure x and y are the same length
    if x.offset > y.offset
        % pad the front of x with zeros
        x.data = [zeros(1, x.offset - y.offset), x.data];

        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
```

```
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end

    elseif x.offset < y.offset
        % pad the front of y with zeros
        y.data = [zeros(1, y.offset - x.offset), y.data];

        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end
    else
        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end
    end

    % subtract the data
    z.data = x.data - y.data;
    % set the offset to the minimum of the two
    z.offset = min(x.offset, y.offset);
    z = trimit(z);

--- multit.m --------------------------------

function z = multit(x, y)
% MULTIT Multiply x and y (i.e. .*) Ether x and y will both be sequence
structures of one of them may be a number.
    if isnumeric(x)
        z.data = x .* y.data;
        z.offset = y.offset;
        z = trimit(z);
        return
    elseif isnumeric(y)
        z.data = x.data .* y;
        z.offset = x.offset;
        z = trimit(z);
        return
    end

    % case where x and y are both sequence structures
    % make sure x and y are the same length
    if x.offset > y.offset
        % pad the front of x with zeros
        x.data = [zeros(1, x.offset - y.offset), x.data];

        if length(x.data) > length(y.data)
```

```
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end

    elseif x.offset < y.offset
        % pad the front of y with zeros
        y.data = [zeros(1, y.offset - x.offset), y.data];

        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end
    else
        if length(x.data) > length(y.data)
            % pad the end of y with zeros
            y.data = [y.data, zeros(1, length(x.data) - length(y.data))];
        elseif length(x.data) < length(y.data)
            % pad the end of x with zeros
            x.data = [x.data, zeros(1, length(y.data) - length(x.data))];
        end
    end

    % add the data
    z.data = x.data .* y.data;
    % set the offset to the minimum of the two
    z.offset = min(x.offset, y.offset);
    z = trimit(z);

--- trimit.m --------------------------------

function z = trimit(x)
% TRIMIT Remove leading and trailing zeros from sequence x and adjust offset
appropriately.
    trim_idx = find(x.data);
    % If x is all zeros, return [0] with offset 0
    if isempty(trim_idx)
        z = x;
        z.data = 0;
        z.offset = 0;
        return
    end

    z.data = x.data(trim_idx(1):trim_idx(end));
    z.offset = x.offset + trim_idx(1) - 1; % -1 because of 1-based indexing

--- stemit.m --------------------------------

function stemit(x)
```

```
% STEMIT Display a Matlab sequence x using a stem plot.
    stem([x.offset:length(x.data) + x.offset - 1], x.data);
```



Published with MATLAB® R2023b