```
% *** Aidan Chin, 9/22/23

% *** Aidan Chin, 11/08/2023, Updated for M6 ***

    %   *** Label carts from left to right as 1, 2, 3
    %   *** Label "collisions" A, B, C, etc. until everything is final
    %   *** do not use loops or functions
    %   *** use arrays to keep track of everything
    %   *** Formulas from M1, http://hyperphysics.phy-astr.gsu.edu/hbase/elacol2.html
    %   *** This code tracks the number of collisions between 3 objects in
    %   *** 1D elastic collision

clear

% ----- Getting Started -----

% givens

m = [ input('input mass of cart 1 (g): ') ...
        input('input mass of cart 2 (g): ') ...
        input('input mass of cart 3 (g): ') ];
        % mass of cars in g from left to right

v = [ input('input velocity of cart 1 (m/s): ') ...
        input('input velocity of cart 2 (m/s): ') ...
        input('input velocity of cart 3 (m/s): ') ];
        % velocity of cars in cm/s from left to right

threshold = input('please specify an error threshold: ');
        % error threshold where simulation will end

% check which cart collides first with cart 2

if abs(v(1)-v(2)) == abs(v(2)+v(3)) % if ambiguous, promps user to specify
    hit = input(['the first collision is ambiguous, please specify which ' ...
        'cart collides with card 2 first: ']);
elseif abs(v(1)-v(2)) > abs(v(2)+v(3)) % cart 1 collides first
    hit = 1; % set cart hit next to 1
elseif abs(v(1)-v(2)) < abs(v(2)+v(3)) % cart 3 collides first
    hit = 3; % set cart hit next to 3
end

% set up total masses for two types of collisions

m23 = m(2) + m(3); % total mass of carts 2 and 3 in g
m12 = m(1) + m(2); % total mass of carts 1 and 2 in g

% set up the checks by computing total energy and momentum

KE0 = sum(.5.*m.*v.^2); %calculates the total kenetic energy initially
P0 = sum(m.*v);         %calculates the total potential energy initially
```

```
52
53 c = 0; % set up counter for number of collisions
54
55 while v(1) > v(2) || v(3) < v(2) % loop if carts continue to collide
56
57     if hit == 3 % checks if carts 2 and 3 will collide
58         fprintf('carts 2 and 3 will collide\n') % state situation
59
60         vR = [ 0 0 0 ];   % initialize vR
61
62         vR(1) = v(1);  %no interaction - velocity of cart 1 remains constant in cm/s
63
64         vR(2) = (m(2)-m(3))/m23*v(2) + 2*m(3)/m23*v(3); %resultant velocity of
65                                                         %cart 2 in cm/s
66         vR(3) = 2*m(2)/m23*v(2) - (m(2)-m(3))/m23*v(3); %resultant velocity of
67                                                         %cart 3 in cm/s
68
69         fprintf(['after collision number %d the resultant velocity vector is' ...
70             ' [ %d %d %d ]\n'], c, vR)
71         %print the collision number and resultant velocity
72
73         % check energy and momentum (expectations)
74
75         KER = sum(.5.*m.*vR.^2);%calculates the total kinetic energy of situation
76         checkKE = KER-KE0; % Should be zero because no kenetic energy
77                                             % is added or removed
78         PR = sum(m.*vR); %calculates the total potential energy of situation
79         checkP_B = PR-P0;          % should be zero because no potential energy
80                                             % added or removed
81
82         % checks if error threshold has been reached
83         if abs(checkP_B) > threshold || abs(checkKE) > threshold
84             fprintf(['threshold has been reached, terminating simulation \n ' ...
85                 'Potential energy check: %d \n Kinetic energy check: %d \n' ...
86                 ],checkP_B, checkKE)
87             break
88             %state the threshold has been reached, and tell user the
89             %simulation is terminated
90         end
91         hit = 1; %set next cart to be hit as 1
92     elseif hit == 1 % checks if carts 1 and 2 will collide
93         fprintf('carts 1 and 2 will collide\n')
94
95         vR = [ 0 0 0 ]; % initialize vR
96
97         vR(1) = (m(1)-m(2))/m12*v(1) + (2*m(2))/m12*v(2); %resultant velocity of
98                                                         %cart 1 in cm/s
99         vR(2) = (2*m(1)/m12)*v(1) - (m(1)-m(2))/m12*v(2); %resultant velocity of
100                                                        %cart 2 in cm/s
101        vR(3) = v(3); %no interaction - velocity of cart 3 remains constant in cm/s
102
```

```
103         fprintf(['after collision number %d the resultant velocity vector is' ...
104             ' [ %d %d %d ]\n'], c, vR)
105         %print the collision number and resultant velocity
106 
107         % check energy and momentum (expectations)
108 
109         KER = sum(.5.*m.*vR.^2);%calculates the total kinetic energy of situation
110         checkKE = KER-KE0; % Should be zero because no kenetic energy
111                                             % is added or removed
112         PR = sum(m.*vR); %calculates the total potential energy of situation
113         checkP_B = PR-P0;           % should be zero because no potential energy
114                                             % added or removed
115 
116         % checks if error threshold has been reached
117         if abs(checkP_B) > threshold || abs(checkKE) > threshold
118             fprintf(['threshold has been reached, terminating simulation \n ' ...
119                 'Potential energy check: %d \n Kinetic energy check: %d \n' ...
120                 ],checkP_B, checkKE)
121             break
122             %state the threshold has been reached, and tell user the
123             %simulation is terminated
124         end
125         hit = 3; %set next cart to be hit as 3
126     end
127     try % in case there are no collisions, vR wont exist, therefore
128         % set a catch condition
129         v = vR;
130         c = c + 1;
131     catch % if there are no collisions just exit loop
132         break
133     end
134 end
135 try %if there are no collisoins, vR wont exist, therefore set a catch condition
136     fprintf(['after all collisions the resultant velocity vector is' ...
137             ' [ %d %d %d ]\nThere are no more collisions\n'], vR)
138 catch % say different message if carts never collide
139     fprintf(['the carts will never collide the resultant velocity vector is' ...
140             ' [ %d %d %d ]\n'], v)
141 end
```