C-14 Solutions

1. Package Design

Data Structures

- An array of nine integers

Subprograms

- Function to accept 9 integers
- Procedure to display the array in row major order
- Procedure to display the array in column major order
- Procedure to sort the array using the bubble sort algorithm

Algorithms

```
Accept_Numbers:
        For I in 1 .. 9
                Accept an integer
                Store it in an array

Row_Major_Display:
        For I in 1 .. 9
                Display Element in Array(I);
                If I mod 3 = 0 then
                        New_Line
```

Given the elements are in row-major order, the position locations are sequential.

```
Column_Major_Display:
        For I in 1 .. 3
                For J in 1 .. 3
                        Location_In_Array := I + (J-1)*3
                        Display Element in Array(Location_In_Array)
                New_Line
```

If the elements are in column-major order, the locations in the one-dimensional array have to be computed.

```
Bubble_Sort:
        For I in 1 .. Array'Value(1)-1
                For J in I+1 .. Array'Value(1)
                        If Array(I) > Array(J)
                                Swap the values
```

Note that the algorithm will sort the array in ascending order.

2. Code Listing

Package Listing

Package Specification

GNAT 3.13p (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Checking: c:/docume~2/joeb/desktop/16070/codeso~1/my_array_package.ads (source file time stamp: 2003-10-08 13:48:58)

```
 1. ----------------------------------------
 2. -- Package specification of a package to
 3. --   1. Create an array of 9 integers
 4. --   2. Display the array as a 3x3 matrix
 5. --       i. assuming row-major order
 6. --      ii. assuming column-major order
 7. --   3. Bubble Sort the 1-D array
 8. --
 9. -- Specifier : Joe B
10. -- Date Last Modified : 10/07/03
11. ----------------------------------------
12.
13. package My_Array_Package is
14.
15.    type My_Array is array (1 .. 9) of Integer;
16.
17.
18.    function Create_Array return My_Array;
19.
20.    procedure Display_Row_Major(Input_Array : in My_Array);
21.
22.
23.    procedure Display_Column_Major(Input_Array : in My_Array);
24.
25.
26.    procedure Bubble_Sort (Input_Array : in out My_Array);
27. end My_Array_Package;
28.
29.
```

29 lines: No errors

Package Body

GNAT 3.13p (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Compiling: c:/docume~2/joeb/desktop/16070/codeso~1/my_array_package.adb (source file time stamp: 2003-10-08 13:54:40)

```
1. ----------------------------------------
2. -- Package implementation of My_Array_Package
3. -- Programmer : Joe B
```

```
4. -- Date Last Modified : 10/07/03
5. ------------------------------------------
6.
7. with Ada.Text_Io;
8. with Ada.Integer_Text_Io;
9.
10. package body My_Array_Package is
11.
12.
13.   function Create_Array return My_Array is
14.     Output_Array : My_Array;
15.   begin
16.     for I in 1 .. 9 loop
17.       Ada.Text_Io.Put("Please Enter a number : ");
18.       Ada.Integer_Text_Io.Get(Output_Array(I));
19.       Ada.Text_Io.New_Line;
20.     end loop;
21.     return Output_Array;
22.   end Create_Array;
23.
24.
25.   procedure Display_Row_Major (
26.       Input_Array : in     My_Array ) is
27.   begin
28.     for I in 1 .. 9 loop
29.       Ada.Text_Io.Put(Integer'Image(Input_Array(I)));
30.       Ada.Text_Io.Put("    ");
31.       if I mod 3 = 0 then
32.         Ada.Text_Io.New_Line;
33.       end if;
34.     end loop;
35.   end Display_Row_Major;
36.
37.
38.
39.   procedure Display_Column_Major (
40.       Input_Array : in     My_Array ) is
41.     Index : Integer;
42.   begin
43.     for I in 1 .. 3 loop
44.       for J in 1.. 3 loop
45.         Index := I + (J-1)*3;
46.         Ada.Text_Io.Put(Integer'Image(Input_Array(Index)));
47.         Ada.Text_Io.Put("    ");
48.       end loop;
49.       Ada.Text_Io.New_Line;
50.     end loop;
51.   end Display_Column_Major;
52.
53.
54.
55.   procedure Bubble_Sort (
56.       Input_Array : in out My_Array ) is
57.     Temp : Integer;
58.   begin
59.     for I in 1 .. 8 loop
60.       for J in I+1 .. 9 loop
61.         if Input_Array(I) > Input_Array(J) then
```

```
62.              Temp := Input_Array(I);
63.              Input_Array(I) := Input_Array(J);
64.              Input_Array(J) := Temp;
65.           end if;
66.        end loop;
67.     end loop;
68.   end Bubble_Sort;
69.
70. end My_Array_Package;
71.
72.
```

72 lines: No errors

Test Program Listing

GNAT 3.13p (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Compiling: c:/docume~2/joeb/desktop/16070/codeso~1/test_my_array.adb (source file time stamp: 2003-10-08 14:03:20)

```
1. ---------------------------------------------------
2. -- Program to test My_Array_Package
3. -- Programmer : Joe B
4. -- Date Last Modified : 10/07/2003
5. ---------------------------------------------------
6.
7. with My_Array_Package;
8. use My_Array_Package;
9. with Ada.Text_Io;
10.
11. procedure Test_My_Array is
12.    New_Array : My_Array_Package.My_Array;
13. begin
14.    New_Array := Create_Array;
15.
16.    Ada.Text_Io.Put_Line("Displaying unsorted array in Row-Major order");
17.    Display_Row_Major(New_Array);
18.
19.    Ada.Text_Io.Put_Line("Displaying unsorted array in Column-Major order");
20.    Display_Column_Major(New_Array);
21.
22.    Bubble_Sort(New_Array);
23.
24.    Ada.Text_Io.Put_Line("Displaying sorted array in Row-Major order");
25.    Display_Row_Major(New_Array);
26.
27.    Ada.Text_Io.Put_Line("Displaying sorted array in Column-Major order");
28.    Display_Column_Major(New_Array);
29. end Test_My_Array;
```

29 lines: No errors

C –15 Solutions

1. Invert a 3x3 Matrix

Data Structure

A new type my_3x3_matrix as a real array (1..3, 1..3)

Subprograms

1. Function to create the matrix
2. Procedure to display the matrix
3. Function to compute the determinant
4. Function to compute the inverse
5. Main program to invert the matrix

Algorithm

Create_Matrix:

```
For I in 1 .. 3
        For J in 1 .. 3
                Accept Matrix(I,J)
Return Matrix to the user
```

Display_Matrix:

```
For I in 1.. 3
        For J in 1.. 3
                Display Matrix (I,J);
        New_Line
```

Compute_Determinant:

1. Convert the matrix into a real_matrix (as defined in Generic_Real_arrays)
2. Compute the determinant using the det function defined in Generic_Real_Arrays.Operations.
3. Return the computed Value

Compute_Inverse:

1. Convert the matrix into a real_matrix (as defined in Generic_Real_arrays)
2. Compute the inverse using the Inverse function defined in Generic_Real_Arrays.Operations.

3. Convert the real_matrix back into the user defined type
4. Return the inverted matrix

Main Program:

1. Prompt the user to enter a matrix
2. Display accepted matrix to the user
3. Check if matrix is singular by computing the determinant.
4. If matrix is not singular (determinant /= 0)
    a. Compute the inverse
    b. Display inverse to the user
5. Else, Display “Cannot Invert”

2. Code Listing

GNAT 3.13p (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Compiling: c:/docume~2/joeb/desktop/16070c~1/matrix/pset_4_inversion.adb (source file time stamp: 2003-10-08 14:37:14)

```
 1. ------------------------------------------------------
 2. -- Program to invert a 3x3 matrix
 3. -- Programmer : Joe B
 4. -- Date Last Modified : 10/07/03
 5. ------------------------------------------------------
 6.
 7. with Ada.Text_Io;
 8. with Ada.Float_Text_Io;
 9. with Generic_Real_Arrays;
10. with Generic_Real_Arrays.Operations;
11. with Generic_Real_Arrays.Array_Io;
12.
13. procedure Pset_4_Inversion is
14.    -- create instances of the generic matrix packages.
15.    package My_Real_Array is new Generic_Real_Arrays(Float);
16.    package My_Real_Array_Operations is new My_Real_Array.Operations;
17.
18.    -- create a user defined 3x3 matrix
19.    type My_3x3_Matrix is new My_Real_Array.Real_Matrix
20.         (1 .. 3, 1 .. 3);
21.
22.    -- declare local variables for matrices and determinant
23.    A,
24.    B   : My_3x3_Matrix;
25.    Det : Float;
26.
27.    -- user function to create the matrix
28.    function Create_My_Matrix return My_3x3_Matrix is
29.       Input_Matrix : My_3x3_Matrix;
30.
31.    begin
32.       for I in 1 .. 3 loop
```

```
33.       for J in 1 .. 3 loop
34.          Ada.Text_Io.Put("Please Enter Number in (");
35.          Ada.Text_Io.Put(Integer'Image(I));
36.          Ada.Text_Io.Put(",");
37.          Ada.Text_Io.Put(Integer'Image(J));
38.          Ada.Text_Io.Put(") : ");
39.
40.          Ada.Float_Text_Io.Get(Input_Matrix(I,J));
41.          Ada.Text_Io.Skip_Line;
42.       end loop;
43.    end loop;
44.    return Input_Matrix;
45. end Create_My_Matrix;
46.
47. -- user procedure to display the matrix
48. procedure Display_My_Matrix (
49.       Input_Matrix : in     My_3x3_Matrix ) is
50.
51. begin
52.    for I in 1 .. 3 loop
53.       for J in 1 .. 3 loop
54.          Ada.Float_Text_Io.Put(Input_Matrix(I,J));
55.       end loop;
56.       Ada.Text_Io.New_Line;
57.    end loop;
58. end Display_My_Matrix;
59.
60. -- function to compute the inverse
61. function My_Inverse (
62.       Input_Matrix : My_3x3_Matrix )
63.   return My_3x3_Matrix is
64.    -- local variable to convert the user defined matrix to the package defined
65.    -- matrix
66.    My_Real_Matrix : My_Real_Array.Real_Matrix (1 .. 3, 1 .. 3);
67.    Output_Matrix  : My_3x3_Matrix;
68.
69. begin
70.    -- do type conversion from user defined type to package defined type
71.    for I in 1.. 3 loop
72.       for J in 1 .. 3 loop
73.          My_Real_Matrix(I,J) := Input_Matrix(I,J);
74.       end loop;
75.    end loop;
76.    -- compute inverse using the generic package function
77.    My_Real_Matrix := My_Real_Array_Operations.Inverse(My_Real_Matrix);
78.    -- reconvert back to user defined type
79.    for I in 1.. 3 loop
80.       for J in 1 .. 3 loop
81.          Output_Matrix(I,J) := My_Real_Matrix(I,J);
82.       end loop;
83.    end loop;
84.    --return computed inverse
85.    return Output_Matrix;
86.
87. end My_Inverse;
88.
89.
90. function My_Determinant (
```

```
91.          Input_Matrix : My_3x3_Matrix )
92.       return Float is
93.        My_Real_Matrix : My_Real_Array.Real_Matrix (1 .. 3, 1 .. 3);
94.     begin
95.        -- do type conversion from user defined type to package type
96.        for I in 1.. 3 loop
97.           for J in 1 .. 3 loop
98.              My_Real_Matrix(I,J) := Input_Matrix(I,J);
99.           end loop;
100.       end loop;
101.       -- compute the determinant and return to user
102.       return (My_Real_Array_Operations.Det(My_Real_Matrix));
103.    end My_Determinant;
104.
105. begin
106.
107.    -- create and display the matrix
108.    Ada.Text_Io.Put_Line("Please Enter the Matrix : ");
109.    A := Create_My_Matrix;
110.
111.    Ada.Text_Io.Put_Line("Created Matrix : ");
112.    Display_My_Matrix(A);
113.
114.    Ada.Text_Io.New_Line;
115.
116.    -- compute determinant
117.    Det := My_Determinant(A);
118.
119.    -- check for singularity
120.    if Det = 0.0 then
121.       Ada.Text_Io.Put_Line("Cannot invert the matrix");
122.    else
123.       -- compute inverse and display
124.       B := My_Inverse(A);
125.
126.       Ada.Text_Io.New_Line;
127.       Ada.Text_Io.Put_Line("Inverted Matrix : ");
128.       Display_My_Matrix(B);
129.    end if;
130.
131. end Pset_4_Inversion;
```

131 lines: No errors

C-16 Solutions

1. Record Declaration

```
type Aircraft_Information is record
  Aircraft_Number : Integer;
  Latitude : Float;
  Longitude : Float;
  Heading : Float;
  Velocity : Float;
end record;
```

Note there are multiple ways to do this record declaration. You may for instance choose to use a hierachical record wherein the position information (Latitude, Longitude, Heading, Velocity) is a record within Aircraft_Information as shown below:

```
type Position_Information is record
  Latitude : Float;
  Longitude : Float;
  Heading : Float;
  Velocity : Float;
end record;

type Aircraft_Information is record
  Aircraft_Number : Integer;
  Aircraft_Position : Position_Information;
end record;
```

2. Ada Program

Data Structures

Array of 10 elements of Type Aircraft_Information

Subprograms

- Function to create the array of aircraft
- Procedure to sort the contents of the array based on latitude
- Procedure to compute and display the distances between the first aircraft and all other aircraft.

Algorithms

Create_Aircraft:
    For I in 1 .. 10
        Prompt the user to input relevant information
        Store the information in Array(I)
    Return Array to the main program

Sort_Aircraft:

    For I in 1 .. Num_of_Aircraft -1
        For J in I+1 .. Num_Of_Aircraft
            If Array(I).Latitude > Array(J).Latitude
                Swap the records in Array(I) and Array(J)
    Return sorted array to the user

Compute_Distances:

    For I in 2 .. Num_Of_Aircraft
        Compute difference in latitudes (dlat)
        Compute the differences in longitude (dlon)
        Covert the differences into distances using the WGS-84
        approximations in the handout (dlat_dist, dlon_dist)
        Distance between the aircraft = sqrt(dlat_dist^2 + dlon_dist^2)
        Display computed distance to the user.

Main Program:
    Create aircraft using the Create_Aircraft function
    Sort the aircraft
    Compute the distances and display it to the user

Code Listing

My_Aircraft Package Specification

```
GNAT 3.13p  (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Checking: c:/docume~2/joeb/desktop/16070/codeso~1/my_aircraft.ads (source file time stamp: 2003-10-08
18:10:40)

 1.
 2. -------------------------------------------------
 3. -- Package to specify aircraft parameters and the
 4. -- related subprograms
 5. -- Specifier : Joe B
 6. -- Date Last Modified : 10/07/03
 7. -------------------------------------------------
 8. package My_Aircraft is
 9.    Num_of_Aircraft : constant Integer := 10;
10.
11.    type Aircraft_Information is
```

```
12.      record
13.         Aircraft_Number : Integer;
14.         Latitude        : Float;
15.         Longitude       : Float;
16.         Heading         : Float;
17.         Velocity        : Float;
18.      end record;
19.
20.   type Aircraft_Array is array (1 .. Num_Of_Aircraft) of Aircraft_Information;
21.
22.   Latitude_Conversion : constant Float := 1852.24;
23.   Longitude_Conversion : constant Float := 1314.13 ;
24.
25.   function Get_Aircraft_Info return Aircraft_Array;
26.
27.   procedure Sort_Aircraft (
28.         Input_Array : in out Aircraft_Array );
29.
30.   procedure Compute_Distances (
31.         Input_Array : in     Aircraft_Array );
32. end My_Aircraft;
```

32 lines: No errors

My_Aircraft Package Body

GNAT 3.13p (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Compiling: c:/docume~2/joeb/desktop/16070/codeso~1/my_aircraft.adb (source file time stamp: 2003-10-08 18:26:26)

```
1.
2. ------------------------------------------------
3. -- Package body for My_Aircraft
4. -- Programmer: Joe B
5. -- Date Last Modified : 10/07/03
6. ------------------------------------------------
7. with Ada.Text_Io;
8. with Ada.Integer_Text_Io;
9. with Ada.Float_Text_Io;
10. with Ada.Numerics.Elementary_Functions;
11.
12. package body my_aircraft is
13.
14.
15.   function Get_Aircraft_Info return Aircraft_Array is
16.      Output_Array : Aircraft_Array;
17.   begin
18.      for I in 1 .. Num_of_Aircraft loop
19.         Ada.Text_Io.Put("Please Enter Information of Aircraft");
20.         Ada.Text_Io.Put(Integer'Image(I));
21.         Ada.Text_Io.New_Line;
22.
23.         Ada.Text_Io.Put("Aircraft Id : ");
24.         Ada.Integer_Text_IO.Get(Output_Array(I).Aircraft_Number);
25.         Ada.Text_Io.Skip_Line;
26.
```

```
27.         Ada.Text_Io.Put("Latitude : ");
28.         Ada.Float_Text_Io.Get(Output_Array(I).Latitude);
29.         Ada.Text_Io.Skip_Line;
30.
31.         Ada.Text_Io.Put("Longitude: ");
32.         Ada.Float_Text_Io.Get(Output_Array(I).Longitude);
33.         Ada.Text_Io.Skip_Line;
34.
35.
36.         Ada.Text_Io.Put("Heading : ");
37.         Ada.Float_Text_Io.Get(Output_Array(I).Heading);
38.         Ada.Text_Io.Skip_Line;
39.
40.         Ada.Text_Io.Put("Velocity: ");
41.         Ada.Float_Text_Io.Get(Output_Array(I).Velocity);
42.         Ada.Text_Io.Skip_Line;
43.      end loop;
44.      return Output_Array;
45.   end Get_Aircraft_Info;
46.
47.   procedure Sort_Aircraft ( Input_Array : in out Aircraft_Array) is
48.         Temp : Aircraft_Information;
49.   begin
50.      for I in 1 .. Num_of_Aircraft-1 loop
51.         for J in I+1 .. Num_Of_Aircraft loop
52.            if Input_Array(I).Latitude > Input_Array(J).Latitude then
53.               Temp := Input_Array(I);
54.               Input_Array(I) := Input_Array(J);
55.               Input_Array(J) := Temp;
56.            end if;
57.         end loop;
58.      end loop;
59.   end Sort_Aircraft;
60.
61.
62.   procedure Compute_Distances (Input_Array : in Aircraft_Array) is
63.      Distance, dlat, dlat_dist, dlon, dlon_dist : Float;
64.   begin
65.      for I in 2 .. Num_Of_Aircraft loop
66.         Dlat := Input_Array(1).Latitude - Input_Array(I).Latitude;
67.
68.         dlat_dist := dlat * 60.0 * latitude_conversion;
69.         dlat_dist := dlat_dist * dlat_dist;
70.
71.         Dlon := Input_Array(1).Longitude - Input_Array(I).Longitude;
72.         Dlon_Dist := Dlon * 60.0 * Longitude_Conversion;
73.         Dlon_Dist := Dlon_Dist * Dlon_Dist;
74.
75.         Distance := Ada.Numerics.Elementary_Functions.Sqrt(Dlat_Dist+Dlon_Dist);
76.
77.         Ada.Text_Io.Put("The distance between Aircraft with id ");
78.         Ada.Text_IO.Put(Integer'Image(Input_Array(1).Aircraft_Number));
79.         Ada.Text_Io.Put(" Aircraft with id ");
80.         Ada.Text_Io.Put(Integer'Image(Input_Array(I).Aircraft_Number));
81.         Ada.Text_Io.Put(" is ");
82.         Ada.Text_Io.Put(Float'Image(Distance));
83.         Ada.Text_Io.New_Line;
84.      end loop;
```

```
85.    end Compute_Distances;
86. end My_Aircraft;
87.
```

87 lines: No errors

Main Program

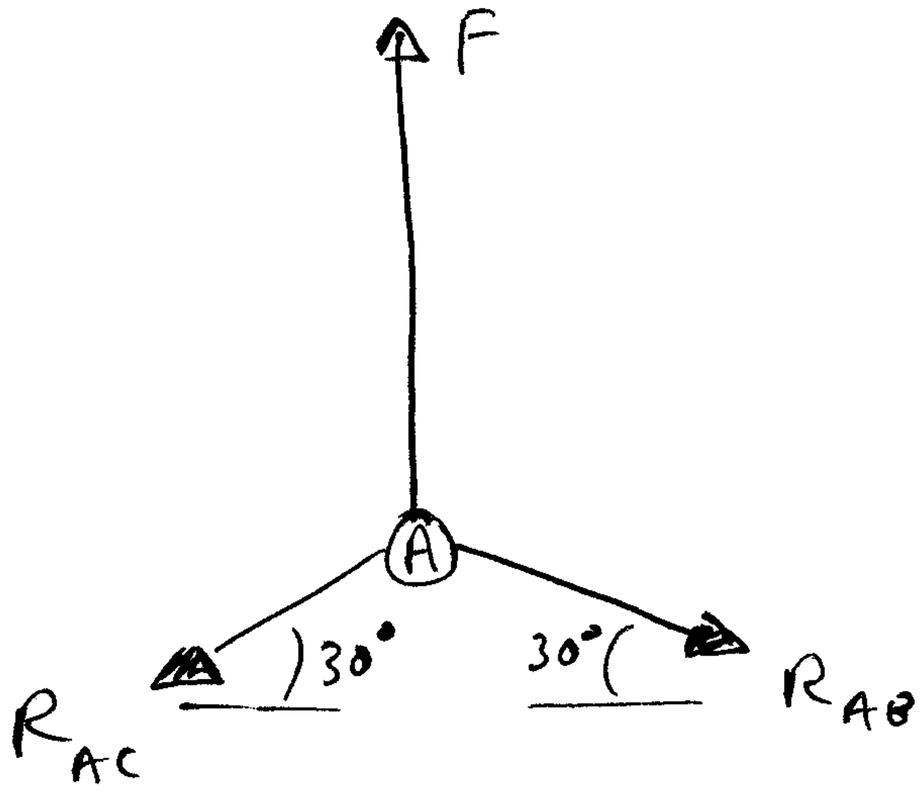
GNAT 3.13p (20000509) Copyright 1992-2000 Free Software Foundation, Inc.

Compiling: c:/docume~2/joeb/desktop/16070/codeso~1/test_my_aircraft.adb (source file time stamp: 2003-10-08 18:12:00)

```
 1. ---------------------------------------------------
 2. -- Program to test My_Aircraft
 3. -- Programmer : Joe B
 4. -- Date Last Modified : 10/07/2003
 5. ---------------------------------------------------
 6.
 7. with My_Aircraft;
 8.
 9. procedure Test_My_Aircraft is
10.    Test_Aircraft : My_Aircraft.Aircraft_Array;
11.
12. begin
13.
14.    Test_Aircraft := My_Aircraft.Get_Aircraft_Info;
15.    My_Aircraft.Sort_Aircraft (Test_Aircraft);
16.    My_Aircraft.Compute_Distances(Test_Aircraft);
17.
18. end Test_My_Aircraft;
19.
20.
```

20 lines: No errors

Problem M1 Solutions



i) By symmetry $R_{AC} = R_{AB} = R$

Apply equilibrium in y dir^{n}

$$F = R_{AC} \sin 30 - R_{AB} \sin 30^\circ = 0$$

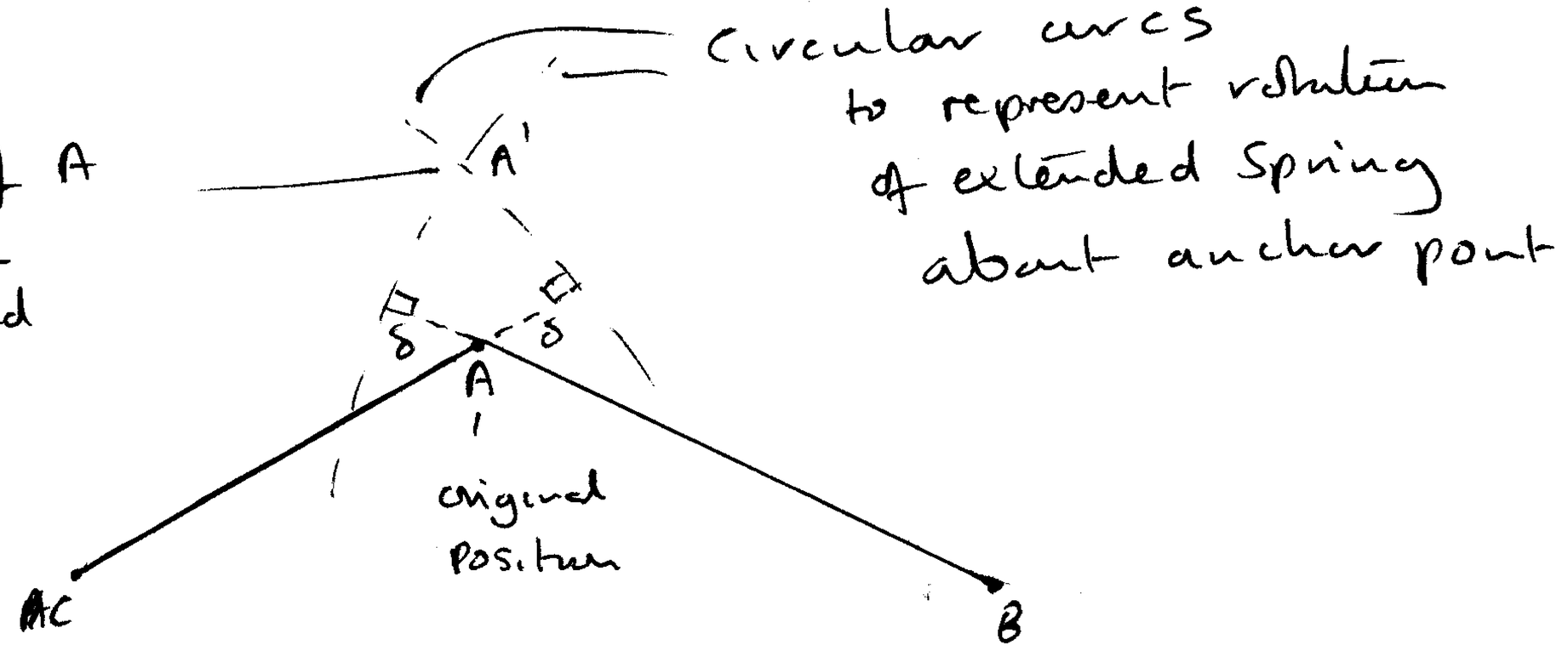
$$F = 2 \times R \times 0.5 = R \quad \Rightarrow \quad R_{AC} = R_{AB} = F \Rightarrow$$

ii) $\therefore \delta_{AC} = \delta_{AB} = \dfrac{R}{K} \quad \Rightarrow$

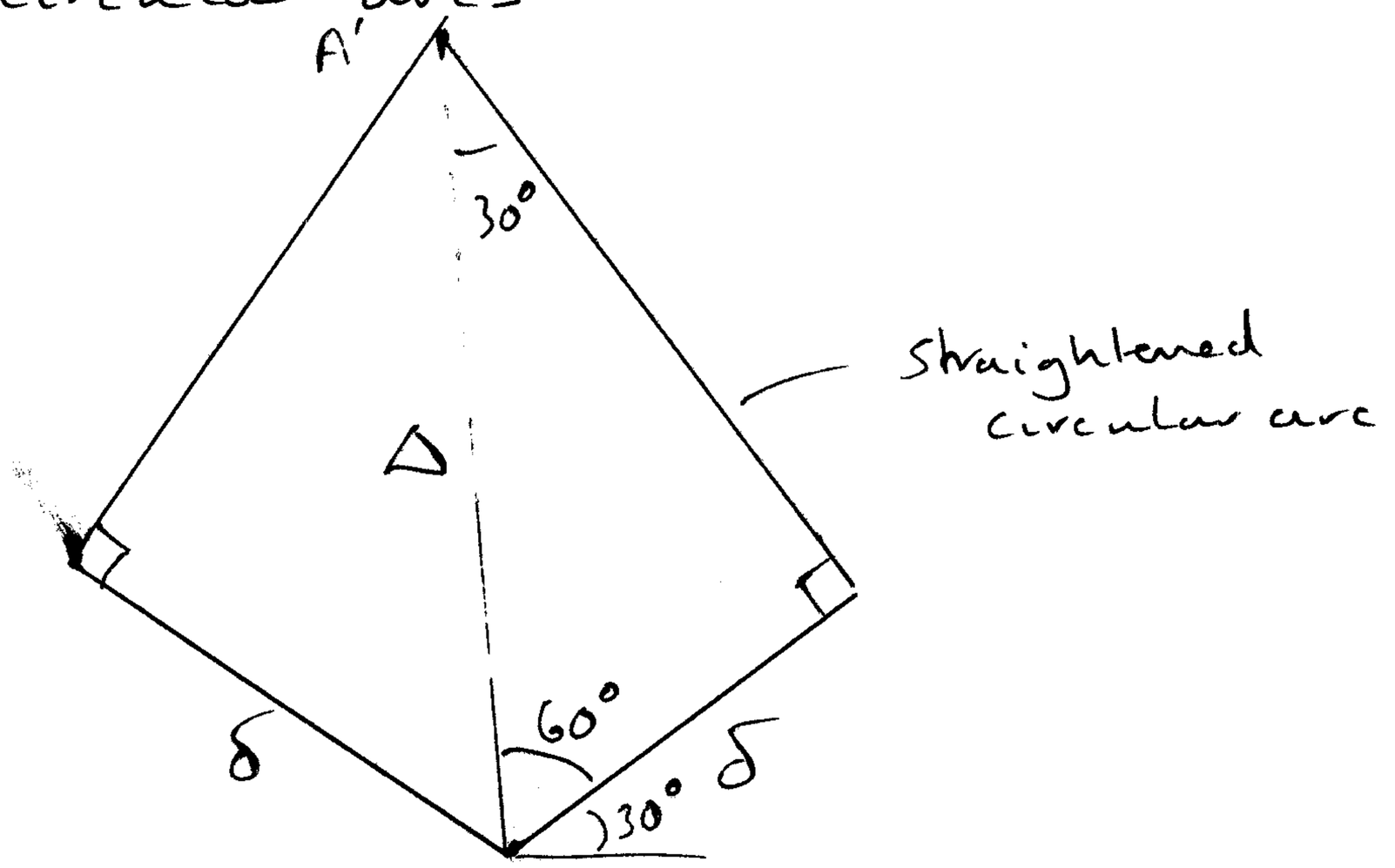
iii) Each spring extends by $\delta = R/K$

Each spring can rotate about its fixed end

Springs remain attached at point A.



Enlarge key region, assume small deflections allow us to ignore circular arcs



A

Vertical displacement of A to A' = Δ

$$\Delta \sin 30^\circ = \delta$$

$$\Delta = 2\delta = \frac{2F}{K} \Longleftarrow$$

(from C.D.L.)

a) $0.4\ \text{hp} \cdot 745.7\ \text{W/hp} = \boxed{298.3\ \text{W}}$

$298.3\ \text{W} / (1.356\ \text{W/ft-lb/s}) = \boxed{220.0\ \text{ft-lb/s}}$

b) heat flow $\dot{H} = 3 \cdot \text{Power} = 894.9\ \text{W}$

heat capacity of water $c = 4.2\ \text{J/g-°K} = 4200\ \text{J/kg-°K}$

typical body mass $m = 70\ \text{kg}$ (155 lb)

rate of temperature increase $\dot{T} = \dfrac{\dot{H}}{mc} = \dfrac{894.9\ \text{W}}{70\ \text{kg} \cdot 4200\ \text{J/kg°K}} = 0.003\ \text{°K/s}$ ($\dot{H}$ in J/s)

Human body can't tolerate more than a few degrees of temperature rise.

Say $\Delta T_{max} = 3\text{°K} = \dot{T}\,\Delta t_{max}$

$\Rightarrow \Delta t_{max} = \dfrac{\Delta T_{max}}{\dot{T}} = \dfrac{3\text{°K}}{0.003\text{°K/s}} = \boxed{1000\ \text{s} = 16.7\ \text{minutes}}$

c) dimensions, using SI units for example:

$\rho \sim \text{kg/m}^3$, $V \sim \text{m/s}$, $S \sim \text{m}^2$, $c \sim \text{m}$

$L \sim N = \text{kg·m/s}^2$ (force), $M \sim \text{N-m} = \text{kg m}^2/\text{s}^2$ (moment)

equation: $L = \frac{1}{2} \rho V^2 S C_L$

units → $\underbrace{\text{kg m/s}^2}_{\text{same}} \sim (\text{kg/m}^3)(\text{m/s})^2 \text{m}^2\, C_L \sim \underbrace{\text{kg m/s}^2}_{\text{same}}\, C_L$

so $\boxed{C_L \text{ is dimensionless}}$

equation: $M = \frac{1}{2} \rho V^2 S c\, C_M$

units → $\text{kg-m}^2/\text{s}^2 \sim (\text{kg/m}^3)(\text{m/s})^2 \text{m}^2 \cdot \text{m}\, C_M \sim \text{kg m}^2/\text{s}^2\, C_M$

so $\boxed{C_M \text{ is dimensionless}}$

d) geometric dimensions scaled by ½, with same airflow

$\rho \to \rho$ same, $V \to V$ same, $S \to \frac{1}{4} S$, $c \to \frac{1}{2} c$, C_L, C_m same

so $\boxed{L \to \frac{1}{4} L}$, $\boxed{M \to \frac{1}{8} M}$