Massachusetts Institute of Technology
Department of Aeronautics and Astronautics
Cambridge, MA 02139

Unified Engineering
Fall 2004

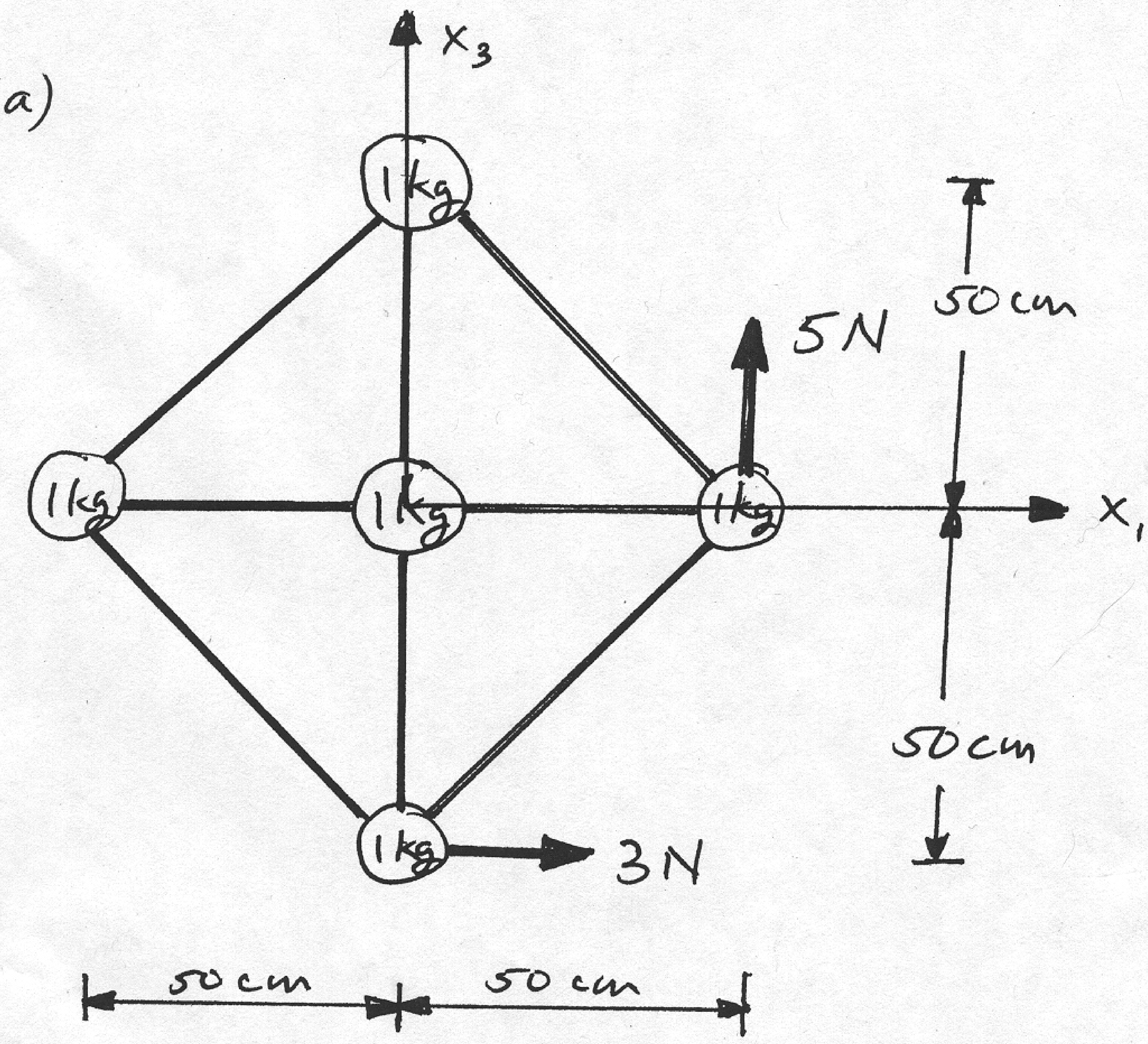
Problem Set #2 Solutions

RdR
9/8/04
(1)

UNIFIED ENGINEERING

Problem Set # 2 - SOLUTIONS

2(M).1 (a)



(b) This system is *not* in equilibrium. One can see that both its net force and net moment are non-zero.

Call the 5N force $\underline{F}_1$ and the 3N force $\underline{F}_2$. Express each as vectors:

$$\underline{F}_1 = 5N\ \hat{i}_3$$
$$\underline{F}_2 = 3N\ \hat{i}_1$$

Thus, the net force, $\underline{F}_N$, is:

$$\underline{F}_N = \underline{F}_1 + \underline{F}_2 = 5N\,\hat{i}_3 + 3N\,\hat{i}_1$$

The acceleration of the system (note that it is rigidly connected) is:

$$\underline{a} = \frac{\underline{F}_{net}}{m}$$

where m is the total mass $= 5 \times \cancel{1}kg = 5\,kg$

So: $$\underline{a} = \frac{5N}{5kg}\hat{i}_3 + \frac{3N}{5kg}\hat{i}_1$$

(NOTE: watch units as $kg = 10^3 g$) and $1N = \frac{kg \cdot m}{s^2}$

So: $$\underline{a} = 1\,m/s^2\,\hat{i}_3 + 0.6\,m/s^2\,\hat{i}_1$$

$\boxed{\text{So the system has a linear acceleration of } 1\,m/s^2 \text{ in the } +x_3 \text{ direction and } 0.6\,m/s^2 \text{ in the } +x_1 \text{ direction}}$

$\boxed{\text{There will also be a counterclockwise rotation about the origin}}$

This is due to the net Moment

$$\underline{M}_{net} = \Sigma(\underline{r}_i \times \underline{F}_i)$$

where:

$\underline{r}_i$ = position vector to force vector from origin one can find:

$$\underline{M}_{net} = (50\text{ cm})\,\hat{i}_1 \times (5N)\,\hat{i}_3 + (-50\text{ cm})\,\hat{i}_3 \times (3N)\,\hat{i}_1$$

Again be wary of units as $\text{cm} = 10^{-2}\text{ m}$

Also noting: $\hat{i}_1 \times \hat{i}_3 = -\hat{i}_2$

$\hat{i}_1 \times \hat{i}_3 = +\hat{i}_2$

$$\Rightarrow \underline{M}_{net} = -2.5\,N\cdot m\;\hat{i}_2 - 1.5\,N\cdot m\;\hat{i}_2$$
$$= (-4.0\,N\cdot m)\,\hat{i}_2$$

(c) By applying other forces and moments, one can achieve equilibrium if the net force and moment are zero.

Thus, by applying a force, one can get the net force to be zero. The applied force must be equal in magnitude and opposite in direction to the current net force (as exists)

$$\Rightarrow \text{apply } \underline{F} = -5N\,\hat{i}_3 - 3N\,\hat{i}_1$$

and the linear acceleration will be zero.

BUT, by applying a force at the origin, one cannot cause a moment about the origin (since $\underline{r} = \underline{0}$), so the net moment cannot become zero

$\Rightarrow \boxed{NO}$ equilibrium cannot be achieved

(d) we have the opposite in this case. By applying a moment about the origin, one can get the net moment to be zero (apply one equal in magnitude, but opposite in direction $\Rightarrow$ $M_{applied} = (4.0\ N\cdot m)\ \hat{i}_2$). However, one cannot get the net force to be zero and thus there will be linear acceleration and motion.

$\Rightarrow \boxed{NO}$ equilibrium cannot be achieved

(e) The same reasoning applies as in the answer to part (d). A couple results in no net force and a net moment. So a couple can cause the rotation to not occur, but the total net force will still be nonzero and thus there will be linear acceleration and motion.

$\Rightarrow \boxed{NO}$ equilibrium cannot be achieved

(f) Go back to the reasonings of parts (c) and (d). Add these two and equilibrium can be achieved

$\Rightarrow \boxed{YES}$

Apply: $\boxed{\begin{array}{l} \underline{F} = -5N\ \hat{i}_3 - 3N\ \hat{i}_1 \\ \underline{M} = (4.0\ N\cdot m)\ \hat{i}_2 \end{array}}$

2(M).2 There are multiple design considerations for all structures and these inevitably involve tradeoffs. The important point here is to identify those considerations which are the most important (i.e. key) to the particular design and thus purpose of that structure. In this vein, there are no "right" answers here since the specific purposes of the structure were not defined. Thus, the first part to answering this question is to define, in your mind, what purpose(s) the structure serves.

It is also important to realize that true tradeoffs (i.e. *relative* importance) can only be done quantitatively when a clear objective is in mind and the factors associated with the different design factors can be quantified. Otherwise, only general statements about tradeoffs can be made. This should become clearer in the "answers" below.

(a) Space station: The structures of a space station serve a number of important purposes. The habitat structure must protect the inhabitants and thus be resistant to meteorite impact. The station will be in operation for many years and thus must have longevity. In orbit, this means resistance to uv, atomic oxygen, fatigue, and other longterm considerations. In orbit, a structure goes in and out of earth's shadow and thus can undergo very large thermal gradients. Thus, dimensional stability is important. Weight and cost are always factors, but it is hard to evaluate these without knowing the scheme for getting the parts of the space station to orbit and also knowing how important the space station is to the country and thus its priority in the budget process.

(b) Commercial transport aircraft: With safety the number one consideration, a key item is strength — ability to carry loads without failure. This also means that there are deformation resistance considerations at various parts of the aircraft. Clearly, the wing must hold its shape to a certain margin otherwise there will be a potential loss in the aerodynamic characteristics and thus the forces on the wing, known as lift and drag. The airplane is designed to last a number of years, so corrosion resistance is important as are considerations of fatigue and general durability. Weight is a clear consideration since this involves a tradeoff with additional payload/fuel/range. Finally, the buyers of the aircraft must be able to afford the aircraft, so cost of the airplane is a key, if not *the* key consideration.

(c) Glider: A glider is also an airplane, but it looks quite different from a commercial transport. Glider wings have a very large "aspect ratio" (length of the wing/"width" of the wing). Due to this great length, glider wings must be very stiff, so deformation resistance is a key consideration with strength being an important design consideration but not as great a consideration as the stiffness/rigidity. Gliders are unpowered and rely on a tow to get to altitude and then on "thermals" to soar higher (Ever watch a hawk or eagle? They do the same thing!). Thus, weight is a much more critical factor with regard to gliders. How about cost? Clearly, cost is a consideration in any consumer product. However, in the sports industry, many people with the "means" are willing to pay extra for extra performance. So cost becomes quite a different factor here since performance becomes a much greater consideration for which people are willing to pay. Thus, cost/performance becomes a key tradeoff.

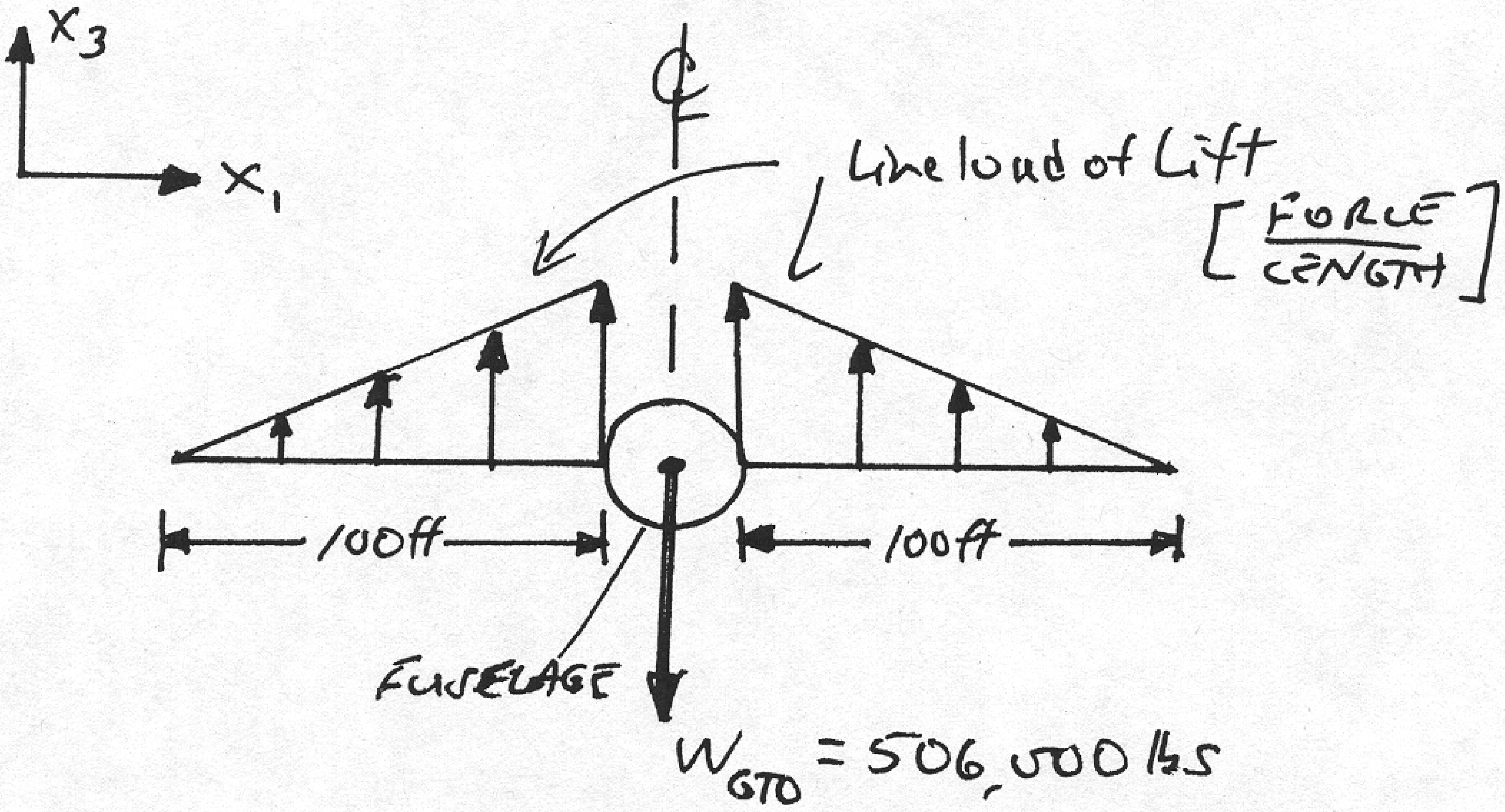
(d) Automobile: The main considerations in an automobile are safety and cost with cost probably being the most important. This is truly a consumer product and the general consumer is much less likely to care about technical innovations and performance capability. They want a product that will work, will get them there, won't break down, and will last. Thus, the next key design consideration is longevity. With cars, especially car bodies, this is inherently linked to corrosion resistance. Another key design consideration is energy absorption in impacts (also known as "crashes"). Much of the body and structure of a car is designed to absorb the energy in such an incident in order to protect the occupant.

(e) Bridge: A bridge is a pretty basic structure and the design considerations are also pretty straightforward. The structure must be strong enough to carry the loads without failure. However, the rigidity of the bridge can be an even more important consideration since deflection/deformation must be resisted (No one wants a "floppy" bridge). Bridges are exposed to the elements, so corrosion resistance is also an important consideration. And this is linked to the final item which is longevity. Bridges are made and used for decades and thus must not fatigue, corrode, etc. This ends up being a tradeoff with the other major consideration — cost. It is a question of the up-front cost of the structure versus the cost "down the road" to maintain and repair. This should include consideration of the inconvenience caused to commuters, etc. when bridges are closed or traffic restricted when major repairs are made (You should have been around when they did this to the Harvard Bridge a few years ago!).

(f) Step ladder: As a consumer product, the most important consideration is cost. However, safety must be right behind this, so this brings in considerations of strength, rigidity, and longevity. Longevity is less important here since it is relatively easy enough to replace this product and one can tell when the material starts to corrode, degrade, etc. Stiffness and strength, however, are clearly important. There are also other considerations. One of the main ones is electrical conductivity. Wood ladders were used for many years and then metal ones were introduced. The problem with metal (generally aluminum) ladders is that if they touch a wire, the person touching the ladder can be electrocuted. Plastic/glass-reinforced ladders have been introduced because of this. They keep down the weight, which is important because people have to be able to carry these things, but are not conductive and provide that extra electrical safety.

2(M).3 (a)

The configuration:



(b) The problem statement gives that the maximum value occurs at the root (placing the origin of the $x_1 - x_3$ system at the root, and assuming the dimensions of the fuselage do not contribute), this occurs at $x_1 = 0$. Now find the magnitude.

<u>Step 1</u>: Each wing must carry half the weight of the airplane through the counteracting action of the force of lift. Lift varies along the wing $\Rightarrow$ Lift $= f(x_1)$

for one wing:

$$\frac{W_{GTO}}{2} = 253{,}000 \text{ lbs} = \int_{root}^{tip} f(x_1)\,dx_1$$

for a 100 ft wing: root = 0 ft; tip = 100 ft

Step 2: Get an expression for the lift as a function of x_1. The lift is a maximum at the root (call this L_R) and goes to zero at the tip with a linear variation. The general expression is:

$$L(x_1) = m x_1 + b$$

First find the slope. This is the ratio of the change in lift to change in distance:

$$m = \frac{(L_R - 0)}{(0\,ft - 100\,ft)} \Rightarrow m = -\frac{L_R}{100\,ft}$$

To find the constant, note that the lift is L_R at the root ($x_1 = 0$). Using this with the derived value for m:

$$L_R = -\frac{L_R}{100\,ft}(0\,ft) + b$$

$$\Rightarrow b = L_R$$

Finally: $$L(x_1) = -\frac{L_R}{100\,ft} x_1 + L_R$$

check this via the other value we know -- at the tip ($x_1 = 100\,ft$), $L = 0$:

$$L(100\,ft) \stackrel{?}{=} -\frac{L_R}{100\,ft}(100\,ft) + L_R$$ ✓ checks

Step 3: Solve using the equations.

$$253{,}000 \text{ lbs} = \int_{0\,ft}^{100\,ft} \left\{\left(-\frac{L_R}{100\,ft}\right) x_1 \quad + L_R\right) dx_1$$

$$\Rightarrow 253{,}000 \text{ lbs} = \left(-\frac{L_R}{100\,ft}\right)\frac{x_1^2}{2}\Bigg]_{0\,ft}^{100\,ft} + L_R x_1 \Bigg]_{0\,ft}^{100\,ft}$$

So:

$$253{,}000 \text{ lbs} = -L_R\, 50\,ft + L_R\, 100\,ft$$

$$= L_R\,(50\,ft)$$

$$\Rightarrow \boxed{L_R = 5060 \text{ lbs/ft}} \text{ at the wing root}$$

NOTE: This is in intensity (lbs/ft) since one must multiply by a length to get a force

(c) An equipollent force system is the net sum of forces and moment on the system.

Start from the expression for the lift line load:

$$L(x_1) = (-50.6 \text{ lbs/ft}^2)\, x_1 + 5060 \text{ lbs/ft}$$

Determining net force and moment require integration:

$$\text{Net force} = \int_{0\,ft}^{100\,ft} L(x_1)\, dx_1$$

$$\text{Net moment}: \int_{0\,ft}^{100\,ft} L(x_1)\, \underbrace{x_1}_{\text{moment arm}}\, dx_1$$

with the moment acting counterclockwise.
Manipulating the expression:

$$F_{net} = \int_{0ft}^{100ft} \left\{ \left(-50.6\ \tfrac{lbs}{ft^2}\right) x_1 + 5060\ \tfrac{lbs}{ft} \right\} dx_1$$

$$= -50.6 \frac{lbs}{ft^2} \frac{x_1^2}{2} + 5060 \frac{lbs}{ft} x_1 \Bigg]_{0ft}^{100ft}$$

$$= -253,000\ lbs + 506,000\ lbs$$

$$\Rightarrow \boxed{F_{net} = 253,000\ lbs}$$

NOTE: Same as in (5) as it must be for equilibrium

$$M_{net} = \int_{0}^{100ft} \left\{ \left(-50.6\ \tfrac{lbs}{ft^2}\right) x_1^2 + 5060\ \tfrac{lbs}{ft}\ x_1 \right\} dx_1$$

$$= -50.6 \frac{lbs}{ft^2} \frac{x_1^3}{3} + 5060 \frac{lbs}{ft} \frac{x_1^2}{2} \Bigg]_{0}^{100ft}$$

$$= -16.87 \times 10^6\ lbs \cdot ft + 25.3 \times 10^6\ lbs \cdot ft$$

$$\Rightarrow \boxed{M_{net} = 8.43 \times 10^6\ lbs \cdot ft}$$

counterclockwise

Summarizing, the equipollent system at the root is:

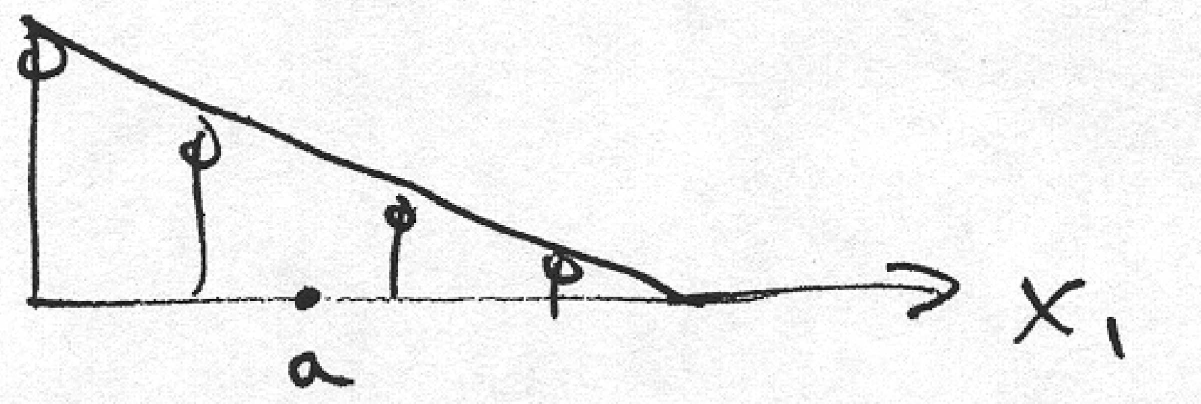
$$\boxed{\begin{array}{ll} +\uparrow & F = 253,000\ lbs \\ \circlearrowleft + & M = 8.43 \times 10^6\ ft \cdot lbs \end{array}}$$

(d) There are two means by which to solve this. First note directly that the net force does not change as the location of the equipollent system changes. Location does not come into play in the integral for force equilibrium.

Method 1: Use the equation

we have already started down this path by reasoning that F_{net} does not change.

One can look at the equation for the net moment and consider any arbitrary location a along the x_1-axis:



M_a = (force due to lift) × (distance from point a)

Expressing this generically and integrating:

$$M_{net_a} = \int_{0ft}^{100ft} \left\{ \left(-50.6 \tfrac{lbs}{ft^2}\right) x_1 + 5060 \tfrac{lbs}{ft} \right\} (x_1 - a)\, dx_1$$

for $\circlearrowleft +$

Perform the calculation:

$$M_{net_a} = \left(-50.6 \tfrac{lbs}{ft^2}\right)\left(\frac{x_1^3}{3} - \frac{a x_1^2}{2}\right) + 5060 \tfrac{lbs}{ft}\left(\frac{x_1^2}{2} - a x_1\right)\Bigg]_{0ft}^{100ft}$$

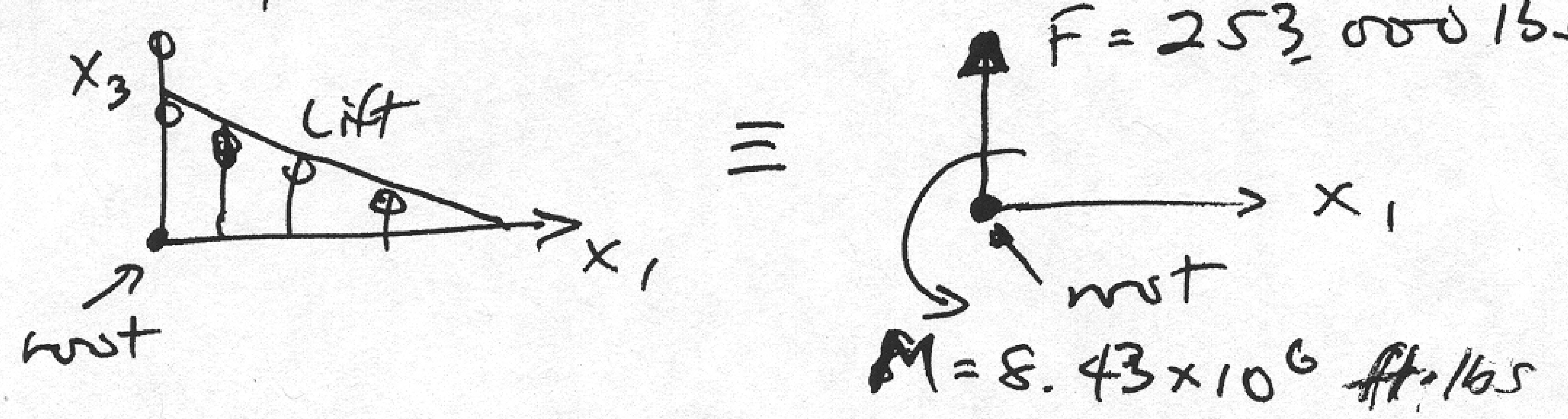
$$\Rightarrow M_{net_a} = -16.87\times10^6 \text{ ft}\cdot\text{lbs} + 0.253a\times10^6 \text{ lbs}$$
$\circlearrowleft +$
$$+25.3\times10^6 \text{ ft}\cdot\text{lbs} - 0.506a\times10^6 \text{ lbs}$$

with a in [ft]:

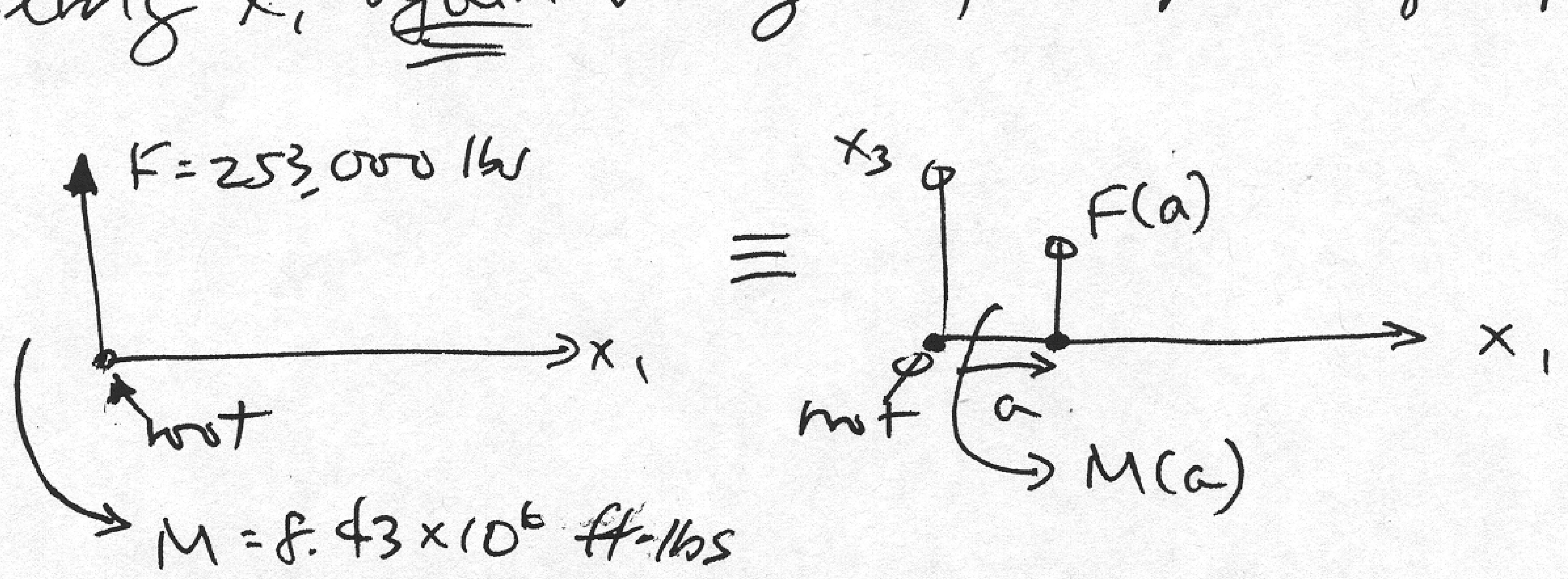
$$M_{net_a} = 8.43\times10^6 \text{ ft}\cdot\text{lbs} - 0.253a\times10^6 \text{ ft}\cdot\text{lbs}$$

and one can plot this

Method 2: use the principle of equipollent system. We can always represent a system of forces (and moments) as an equipollent force and moment. In part (c), we found this acting at/about the root of the system.



We can now take the equipollent pair and move them to see how the force and moment change along x_1 again using the principle of equipollence



Equipollence says:

1. Forces sum the same

$$\Rightarrow \boxed{F = 253{,}000 \text{ lbs.} = F(a)}$$

Does not change (as before)

2. Moments sum to be the same:

$$M = 8.43\times10^{6} \text{ ft}\cdot\text{lbs} = M(a) + F(a)\,a$$

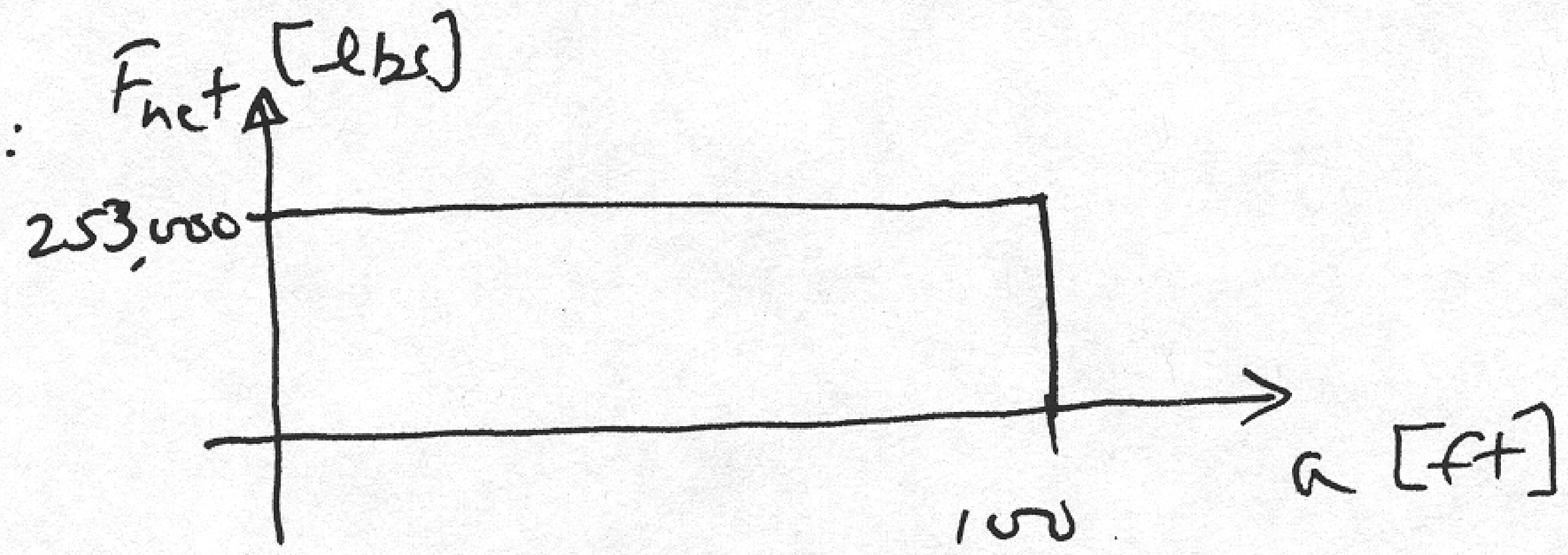
Force being moved creates moment about original point

$$\Rightarrow \boxed{M(a) = 8.43\times10^{6} \text{ ft}\cdot\text{lbs} - 253{,}000 \text{ lbs } (a)}$$

$\circlearrowright +$

Same as via Method 1

Plot:



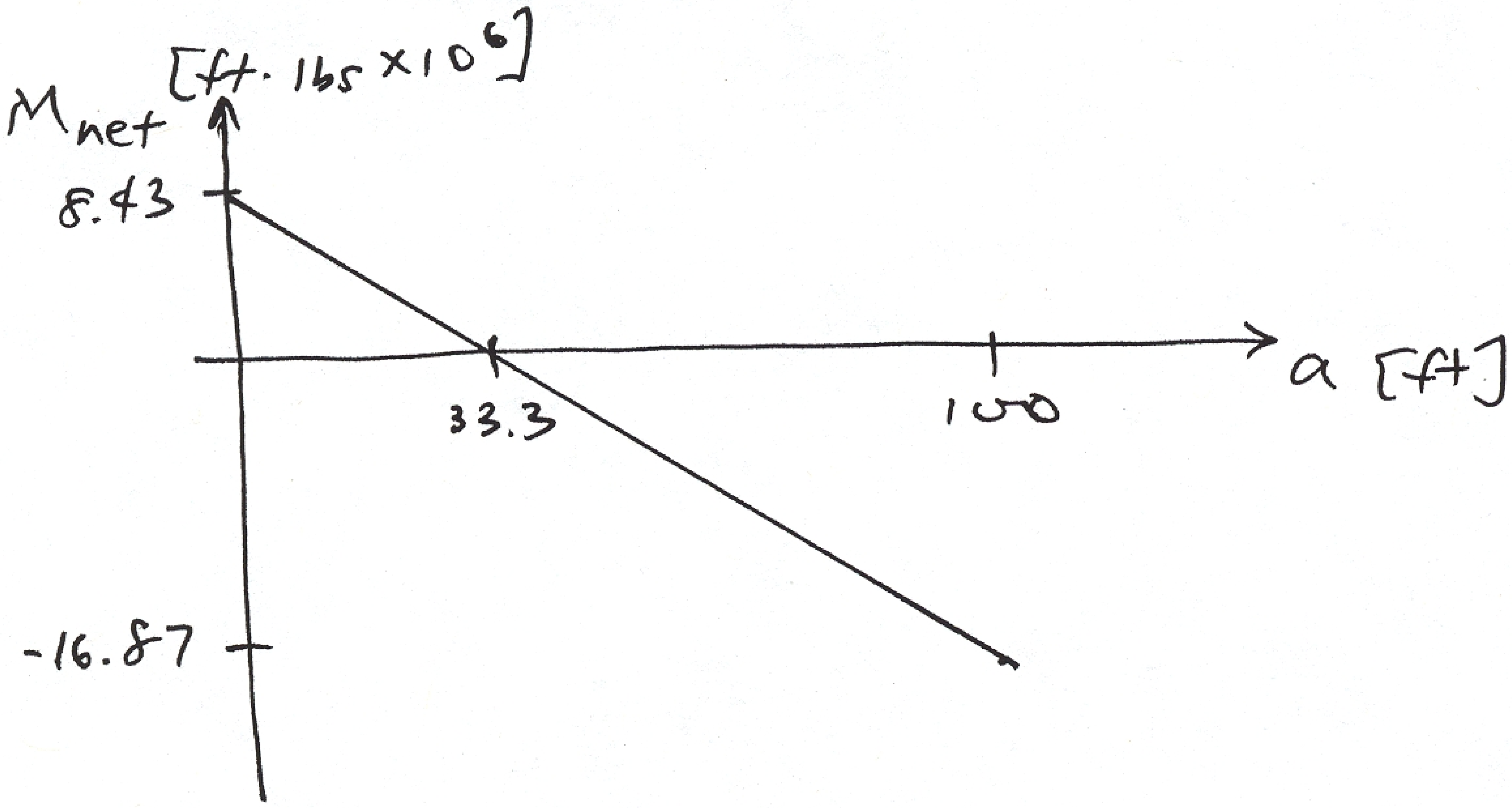
for M_{net} note 3 key points:

at root ($a = 0$ ft) $M = 8.43\times10^{6}$ ft·lbs

at tip ($a = 100$ ft) $M = -16.87\times10^{6}$ ft·lbs

find crossover point where $M = 0$

$$\Rightarrow a = 33.3 \text{ ft}$$


M_{net} [ft·lbs x10^6]
8.43
-16.87
33.3
100
a [ft]

CP2

Problem 1 **10 points**

The Grading rubric is as follows:

1. **5 points if program executes correctly and displays the students name**
2. **2 points for turning in the code listing**
3. **2 point for highlighting their changes**
4. **1 point for comments**

GNAT 3.15p (20020523) Copyright 1992-2002 Free Software Foundation, Inc.

Compiling: c:/docume~1/kristina/psets/pset2/my_first_hello_world.adb (source file time stamp: 2004-09-17 22:48:22)

```
----------------------------------------------------
-- Program to display hello world on the screen
-- Programmer: Ada_Newbie
-- Modified By: Jane B
-- Date Created: September 07, 2004
-- Date Last Modified: September 17, 2004
----------------------------------------------------

with Ada.Text_Io;--specifies the usage of text_io

procedure My_First_Hello_World is
begin
   Ada.Text_Io.Put(" HELLO WORLD!");--displays hello world on the screen
   Ada.Text_Io.New_Line; --moves cursor to the next line
   --this line is added
   Ada.Text_Io.Put("My Name Is Jane B");

end My_First_Hello_World;

```

19 lines: No errors

Problem 2. 40 points

Build Rover based on handout

The rover is graded as follows:

1. Rover works when the demo program is downloaded to it (the demo program can be found on the class website). 30 points

2. Turning in the wiring diagram - 5 points

3. All three team members turning in individual times 5 points

Problem 3. 20 points

The grading rubic is as follows:
1. For identifying that the output_off statement causes the motor to remain on during the turn 5 points

2. For detailing the rover behavior as shown below 15 points
What we are looking for is a detailed explanation of the rover behavior. It need not be represented as a list.

When the line Output_Off(Output => Drive); is commented out, the calls to Forward results in the motor **being left on**. The behavior of the rover is as follows:

a. Goes forward for 2 seconds
b. Arcs left as the steer motor is turned on
c. Moves forward for 0.5 seconds after the steering motor is turned off
d. Steering motor centers the front of the rover
e. Rover moves back for 2 seconds
f. Stops
g. Turns steering right
h. Moves back for 0.5 seconds
i. Centers the steering
j. Moves forward for 2 seconds
k. Rover **does not** stop. → **5 points**
→ **10 points** for the rest

Problem 4. 30 points

While there problem statement did not explicitly ask for an algorithm, the algorithm is included in the solution for the students to understand the format. We will be grading the algorithms from the next problem set.

The program itself is graded based on:

1. Updated program header, comments	**4 points**
2. Rover moves north for 5 seconds	**2 points**
3. Rover turns west	
Turn left	**3 points**
Move forward for 2.5 seconds	**3 points**
Rover moves west for 5 seconds	**2 points**

Similarly 8 points each for south and east

Preconditions: Robot is facing north.
Inputs: None
Outputs: None
Postconditions: Rover back to where it started.

Algorithm:

1. Move the rover forward for 5 seconds. using Forward(50)

2. Turn West
   a. Turn steering left using Turn(Left)
   b. Turn on drive motor power for 2.5 seconds using Forward(25)

3. Move rover forward west for 5 seconds using Forward(50)

4. Turn South
   a. Turn steering left using Turn(Left)
   b. Turn on drive motor power for 2.5 seconds using Forward(25)

5. Move rover forward south for 5 seconds using Forward(50)

6. Turn East
   a. Turn steering left using Turn(Left)
   b. Turn on drive motor power for 2.5 seconds using Forward(25)

7. Move rover forward East for 5 seconds using Forward(50)

Code Listing:

GNAT 3.15p (20020523) Copyright 1992-2002 Free Software Foundation, Inc.

Compiling: c:/docume~1/kristina/psets/pset2/basic_mindstorms_demo.adb (source file time stamp: 2004-09-18 12:45:02)

```
-- A double hyphen indicates the start of a comment. the comments
-- ends at the end of the line. Comments are ignored by the Ada
-- compiler.

with Lego;  -- Informs compiler that this program will be making use of
-- a package called Lego. This package contains Lego related subprograms
-- and types that we need to use

use Lego;

------------------------------------------------------------------
--| Name of file: demo_basic_rover.adb
--| Unified.C&P Lecture #2
--|
--| Program to demo the basic functionality of the Lego Mindstorms rover
--| Original Programmer: Jane B
--| Modified by: Joe B
--| Date Created: September 01, 2004
--| Date Last Modified: September 17, 2004
--|
--| Comments:
--| 1) It is important to note that HOW the rover is wired makes
--| a big difference. I.e., even if "Motor A" is connected to RCX output
--| "A," the orientation of how the terminals of the wire is connected
--| (if it is rotated by 90 degrees) will change the motors rotation
--| direction.
--| 2) Make sure the robot is calibrated: make sure the switch that
--| represents "centered" is triggered and that the steering column
--| is actually centered. If, for some reason the steering column is
--| centered and the touch sensor isn't triggered, the program will
--| attempt to center the robot by operating the turning motor until
--| the touch sensor is triggered.
------------------------------------------------------------------

procedure Demo_Basic_Rover is -- program heading (start of the program)
          |
>>> warning: file name does not match unit name, should be "demo_basic_rover.adb"


   -- Defines Left as a constant object with value 0, and Right to be 1
   -- The value of Left/Right cannot be changed by any subsequent
   -- program statements
   Left  : constant Integer := 0;
   Right : constant Integer := 1;
   |
>>> warning: constant "Right" is not referenced


   -- Defines the Drive motor power source to be Output_A
   Drive : constant Output_Port := Output_A;
   -- Defines the Steering motor power source to be Output_B
   Steer : constant Output_Port := Output_B;

   -- Defines straight to be a sensor port, corretaled to Sensor_1
   Straight : constant Sensor_Port := Sensor_1;
   -- Defines Bumper to be a sensor port, corretaled to Sensor_2
```

```
Bumper : constant Sensor_Port := Sensor_2;

-------------------------------------------------------------------
-- Steer_Left: procedure to steer the front wheels of the rover
-- to the left
-- inputs:  none
-- outputs: none
-- postconditions: rover's front wheels are turned left
procedure Steer_Left is
begin
   --power the steering motor forward
   Output_On_Forward(Output => Steer);
   Wait(Hundredths_Of_A_Second => 25); -- wait for 0,25 seconds
   Output_Off(Output => Steer); -- turn off power to steering motor
end Steer_Left;

-------------------------------------------------------------------
-- Steer_Right: procedure to steer the front wheels of the rover
-- to the right
-- inputs:  none
-- outputs: none
-- postconditions: rover's front wheels are turned right
procedure Steer_Right is
begin
   --power the steering motor backwards
   Output_On_Reverse(Output => Steer);
   Wait(Hundredths_Of_A_Second => 25); -- wait for 0,25 seconds
   Output_Off(Output => Steer); --turn off power to steering motor
end Steer_Right;

-------------------------------------------------------------------
-- Steer_Center: procedure to straighten the rover front wheels
-- inputs:  none
-- outputs: none
-- postconditions: rover front wheels are aligned straight
procedure Steer_Center is
begin
   --change the direction in which steering motor rotates
   Set_Output_Direction(
      Output    => Steer,
      Direction => Output_Direction_Toggle);
   --turn the steering power on
   Output_On(Output => Steer);
   --use the straight sensor input to center the drive
   while (Get_Sensor_Value(Sensor=>Straight)=0) loop
      Wait(Hundredths_Of_A_Second => 1);
   end loop;

   Output_Off(Output => Steer); --turn off power to steering motor
end Steer_Center;

-------------------------------------------------------------------
```

```
-- Go_Forward: procedure to move the rover forward
-- inputs:  none
-- outputs: none
-- postconditions: rover moves forward
procedure Go_Forward is
          |
>>> warning: procedure "Go_Forward" is not referenced

begin
   --turn on drive motor power
   Output_On(Output => Drive);
end Go_Forward;


-----------------------------------------------------------------
-- Forward: procedure to move the rover forward for a given duration
--          of time
-- inputs:  amount of time (in 10th of a second) the rover
--          moves forward
-- outputs: none
-- postconditions: rover moves forward for the given duration of time
procedure Forward (
      Tenths_Of_A_Second : in     Integer ) is
begin
   -- provide power to the drive motor
   Output_On_Forward(Output => Drive);
   -- wait for required duration of time
   Wait(Hundredths_Of_A_Second => Tenths_Of_A_Second * 10);
   --turn off the power to the drive motor
   Output_Off(Output => Drive);
end Forward;


-----------------------------------------------------------------
-- Back: procedure to move the rover backward for a given duration
--       of time
-- inputs:  amount of time (in 10th of a second) the rover moves back
-- outputs: none
-- postconditions: rover moves back for the given duration of time
procedure Back (
          |
>>> warning: procedure "Back" is not referenced

      Tenths_Of_A_Second : in     Integer ) is
begin
   --provide power to the drive motor in reverse
   Output_On_Reverse(Output => Drive);
   -- wait for the required time
   Wait(Hundredths_Of_A_Second => Tenths_Of_A_Second * 10);
   -- turn off power to the drive motor
   Output_Off(Output => Drive);
end Back;


-----------------------------------------------------------------
-- Turn: procedure to turn the rover front wheels in a specified
```

```
--      direction
-- inputs:  direction in which the rover has to turn
-- outputs: none
-- postconditions: rover front wheels in the required direction
procedure Turn (
    Direction : in    Integer ) is
begin

  if Direction = Left then  -- if the direction is to the left
    Steer_Center;         -- center the front wheels
    Steer_Left;           -- turn the wheels left
  else -- turn right
    Steer_Center;
    Steer_Right;
  end if;
end Turn;


-------------------------------------------------------------------
-- Initialize_Rover: procedure to configure the rover
-- inputs:     none
-- outputs:    none
-- postconditions: the sensors on the rover correlated to input and
--              output ports
procedure Initialize_Rover is
begin
  --defines the Straight sensor port to accept Touch Sensor input
  Config_Sensor(
    Sensor => Straight,
    Config => Config_Touch);

  --defines the Bumper sensor port to accept Touch Sensor input
  Config_Sensor(
    Sensor => Bumper,
    Config => Config_Touch);

  --set drive motor power to High
  Output_Power(
    Output => Drive,
    Power  => Power_High);

  --set steering motor power to Low
  Output_Power(
    Output => Steer,
    Power  => Power_Low);

  --align the rover to the center
  Steer_Center;
end Initialize_Rover;




-------------------------------------------------------------------
-- This is the only part of the code that has been changed
-------------------------------------------------------------------
```

```
begin

   Initialize_Rover;    -- initialize the rover
   -- assume that the rover is facing north
   -- move the rover forward for 50*0.1 seconds
   Forward(50);
   --north to west involves a left turn
   --turn the rover steering to the left
   Turn(Left);
   --turn the rover drive motor on for 2.5 seconds
   --to get a 90 degree turn
   Forward(25);
   --center the steering
   Steer_Center;

   -- move the rover forward for 5 seconds
   Forward(50);
   --west to south involves a left turn
   --turn the rover steering to the left
   Turn(Left);
   --turn the rover drive motor on for 2.5 seconds
   --to get a 90 degree turn
   Forward(25);
   --center the steering
   Steer_Center;

   -- move the rover forward for 5 seconds
   Forward(50);
   --south to east involves a left turn
   --turn the rover steering to the left
   Turn(Left);
   --turn the rover drive motor on for 2.5 seconds
   --to get a 90 degree turn
   Forward(25);
   --center the steering
   Steer_Center;

   -- move the rover forward for 5 seconds
   Forward(50);
   --east to north involves a left turn
   --turn the rover steering to the left
   Turn(Left);
   --turn the rover drive motor on for 2.5 seconds
   Forward(25);
   --center the steering
   Steer_Center;

end Demo_Basic_Rover;

```

261 lines: No errors, 4 warnings