C8-1

Algorithm

1. Use a subtype to represent the numbers for months
2. Use an enumeration to represent the named months
3. Use an enumeration to represent the roman months
4. Get the inputs from the user
5. Convert the month into roman and named formats using
    a. New_Type_Package'Val(Month_Type'Pos(Month) –1);
6. Display the months in all three formats to the user.

Note: The enumerations range from 0 to (number_of_elements_in_Enumeration –1).

Code Listing

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

Compiling: c:/docume~2/jk/desktop/16070/codeso~1/translate_dates.adb (source file time stamp: 2003-09-24 01:35:00)

```
 1. ------------------------------------------------------------------
 2. -- Program to accept different date formats
 3. -- Programmer : Jayakanth Srinivasan
 4. -- Date Last Modified : 09-23-2003
 5. ------------------------------------------------------------------
 6.
 7. with Ada.Text_Io;
 8. with Ada.Integer_Text_Io;
 9.
10. procedure Translate_Dates is
11.    --use a subtype to limit the date to be between 1 and 31.
12.    subtype Date_Type is Integer range 1..31;
13.    -- use a subtype to limit the month to be between 1 and 12
14.    subtype Month_Type is Integer range 1..12;
15.
16.    -- define an enumeration type for roman months
17.    type Roman_Month_Type is
18.         (I,
19.          Ii,
20.          Iii,
21.          Iv,
22.          V,
23.          Vi,
24.          Vii,
25.          Viii,
26.          Ix,
27.          X,
28.          Xi,
29.          Xii);
30.
31.    -- define an enumeration type for names of months
32.    type Named_Month_Type is
```

```
33.         (January,
34.          February,
35.          March,
36.          April,
37.          May,
38.          June,
39.          July,
40.          August,
41.          September,
42.          October,
43.          November,
44.          December);
45.
46.    -- create a package that can print the month in roman numerals
47.    package Roman_Month_Io is new Ada.Text_Io.Enumeration_Io(Enum => Roman_Month_Type);
48.
49.    -- create a package that can print the month's name
50.    package Named_Month_Io is new Ada.Text_Io.Enumeration_Io(Enum => Named_Month_Type);
51.
52.    Year       : Integer;
53.    Date       : Date_Type;
54.    Month      : Month_Type;
55.    Roman_Month : Roman_Month_Type;
56.    Named_Month : Named_Month_Type;
57.
58. begin
59.    Ada.Text_Io.Put("Please Enter the Date 1..31 : ");
60.    Ada.Integer_Text_Io.Get(Date);
61.    Ada.Text_Io.New_Line;
62.
63.    Ada.Text_Io.Put("PLease Enter the Month 1 ..12 : ");
64.    Ada.Integer_Text_Io.Get(Month);
65.    Ada.Text_Io.New_Line;
66.
67.    Ada.Text_Io.Put("PLease Enter the Year: ");
68.    Ada.Integer_Text_Io.Get(Year);
69.    Ada.Text_Io.New_Line;
70.
71.    -- convert the month into roman month
72.    Roman_Month := Roman_Month_Type'Val(Month_Type'Pos(Month)-1);
73.
74.    -- convert the month into the name
75.    Named_Month := Named_Month_Type'Val(Month_Type'Pos(Month)-1);
76.
77.    -- display the dates in regular format
78.    Ada.Text_Io.Put(" i. ");
79.    Ada.Integer_Text_Io.Put(Date);
80.    Ada.Text_Io.Put(" / ");
81.    Ada.Integer_Text_Io.Put(Month);
82.    Ada.Text_Io.Put(" / ");
83.    Ada.Integer_Text_Io.Put(Year);
84.    Ada.Text_Io.New_Line;
85.
86.    -- display the date in named format
87.    Ada.Text_Io.Put(" ii. ");
88.    Ada.Integer_Text_Io.Put(Date);
89.    Ada.Text_Io.Put(" ");
90.    Named_Month_Io.Put(Named_Month);
```

```
91.    Ada.Text_Io.Put(" ");
92.    Ada.Integer_Text_Io.Put(Year);
93.    Ada.Text_Io.New_Line;
94.
95.    -- display the date in roman format
96.    Ada.Text_Io.Put(" iii. ");
97.    Ada.Integer_Text_Io.Put(Date);
98.    Ada.Text_Io.Put(".");
99.    Roman_Month_Io.Put(Roman_Month);
100.    Ada.Text_Io.Put(".");
101.    Ada.Integer_Text_Io.Put(Year);
102.    Ada.Text_Io.New_Line;
103.
104. end Translate_Dates;

104 lines: No errors
```

C-8 2. What are the First and Last values of the following data types

a. Integer

Integer'First = -2147483648
Integer'Last = 2147483647

b. Float

Float'First = -3.40282E+38
Float'Last = 3.40282E+38

c. Character

Character'First =
Character'Last =

Note that both the character values are control character and hence do not get printed on the screen. The position values are 0, 255

d. Boolean

Boolean'First = FALSE
Boolean'Last = TRUE

C9-1

Algorithm

Package Specification

1. Declare the type.
2. Create a new package within the package specification to provide predefined functions for the enumeration type.
3. Declare the function prototypes for both the predecessor and successor functions.

Package Implementation

1. Successor function:
    a. if the input_value = Type'Last then
        return Type'First
    b. else
        return Type'Succ(input_value);

2. Predecessor function:
    a. if the input_value = Type'First then
        return Type'Last
    b. else
        return Type'Pred(input_value);

Code Listing

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

Checking: c:/docume~2/jk/desktop/16070/codeso~1/my_type_package.ads (source file time stamp: 2003-09-24 01:59:20)

```
1. ------------------------------------------------------------
2. -- Package specified to declare the type and two functions
3. -- Specifier : Jayakanth Srinivasan
4. -- Date Last Modified : 09/23/2003
5. ------------------------------------------------------------
6.
7. with Ada.Text_Io;
8.
9. package My_Type_Package is
10.
11.    type Day is
12.         (Monday,
13.          Tuesday,
14.          Wednesday,
15.          Thursday,
16.          Friday,
17.          Saturday,
18.          Sunday);
```

```
19.    package Day_Io is new Ada.Text_Io.Enumeration_Io(Enum => Day);
20.
21.    function Successor (
22.          Day_In : Day )
23.      return Day;
24.
25.    function Predecessor (
26.          Day_In : Day )
27.      return Day;
28. end My_Type_Package;
```

28 lines: No errors

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

Compiling: c:/docume~2/jk/desktop/16070/codeso~1/my_type_package.adb (source file time stamp: 2003-09-24 02:00:04)

```
 1. ---------------------------------------------------------
 2. -- Package implementation of My_type package
 3. -- Implementer : Jayakanth Srinivasan
 4. -- Date Last Modified : 09/23/2003
 5. ---------------------------------------------------------
 6.
 7. package body My_Type_Package is
 8.
 9.
10.    function Successor (
11.          Day_In : Day )
12.      return Day is
13.    begin
14.       if Day_In = Day'Last then
15.          return Day'First;
16.       else
17.          return Day'Succ(Day_In);
18.       end if;
19.    end Successor;
20.
21.
22.    function Predecessor (
23.          Day_In : Day )
24.      return Day is
25.    begin
26.       if Day_In = Day'First then
27.          return Day'Last;
28.       else
29.          return Day'Pred(Day_In);
30.       end if;
31.    end Predecessor;
32. end My_Type_Package;
```

32 lines: No errors

C9-2

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

Compiling: c:/docume~2/jk/desktop/16070/codeso~1/test_types.adb (source file time stamp: 2003-09-24 02:03:20)

```
 1. ---------------------------------------------------------
 2. -- Program to test the package implementation
 3. -- Programmer : Jayakanth Srinivasan
 4. -- Date Last Modified : 09/23/2003
 5. ---------------------------------------------------------
 6. with My_Type_Package;
 7. with Ada.Text_Io;
 8.
 9. procedure Test_Types is
10.    My_Day : My_Type_Package.Day;
11.
12. begin
13.    -- initialize the day to monday
14.    My_Day := My_Type_Package.Monday;
15.
16.    My_Type_Package.Day_Io.Put(My_Type_Package.Successor(My_Day));
17.    Ada.Text_Io.New_Line;
18.
19.    My_Type_Package.Day_Io.Put(My_Type_Package.Predecessor(My_Day));
20.    Ada.Text_Io.New_Line;
21.
22.    -- change the day to sunday
23.    My_Day := My_Type_Package.Sunday;
24.
25.    My_Type_Package.Day_Io.Put(My_Type_Package.Successor(My_Day));
26.    Ada.Text_Io.New_Line;
27.
28.    My_Type_Package.Day_Io.Put(My_Type_Package.Predecessor(My_Day));
29.    Ada.Text_Io.New_Line;
30. end Test_Types;
31.
32.
```

32 lines: No errors

C-10 1.
For X = 3

Case 1.

```
IF x >= 0 THEN
      x:= x+1;
ELSIF x >=1 THEN
      x := x + 2;
END IF;
```

In the case above, only the x:= x+1 statement is executed and the result is 4;

Case 2.

```
IF x >= 0 THEN
      x := x + 1;
END IF;
IF x >= 1 THEN
      x := x + 2;
END IF;
```

In this case, both the x:=x+1 and x:= x+2 statements will be executed and the result is 6.

C-10 2.

Package Specification Listing

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

Checking: c:/docume~2/jk/desktop/16070/codeso~1/my_math_package.ads (source file time stamp: 2003-09-24 03:27:46)

```
 1. ------------------------------------------------------------
 2. -- Package specified to implement two arithmetic functions
 3. -- Specifier : Jayakanth Srinivasan
 4. -- Date Last Modified : 09/23/2003
 5. ------------------------------------------------------------
 6.
 7.
 8. package My_Math_Package is
 9.    subtype Menu_Choice is Integer range 1 .. 3;
10.
11.    procedure Menu (
12.          My_Menu_Choice :    out Menu_Choice );
13.
14.    function Add (
15.          X : Float;
16.          Y : Float  )
17.      return Float;
18.
19.    function Multiply (
20.          X : Integer;
21.          Y : Integer  )
22.      return Integer;
23. end My_Math_Package;
```

23 lines: No errors

Package Code Listing

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

Compiling: c:/docume~2/jk/desktop/16070/codeso~1/my_math_package.adb (source file time stamp: 2003-09-24 03:27:46)

```
 1. ------------------------------------------------------------
 2. -- Package implementation of My_Math package
 3. -- Implementer : Jayakanth Srinivasan
 4. -- Date Last Modified : 09/23/2003
 5. ------------------------------------------------------------
 6. with Ada.Integer_Text_Io;
 7. with Ada.Text_Io;
 8. with Ada.Float_Text_Io;
          |
 9.
```

```
10. package body My_Math_Package is
11.
12.
13.    function Add (
14.          X : float;
15.          Y : float)
16.     return float is
17.    begin
18.       return (X+Y);
19.
20.    end Add;
21.
22.
23.    function Multiply (
24.          X : Integer;
25.          Y : Integer  )
26.     return Integer is
27.    begin
28.       return (X*Y);
29.    end Multiply;
30.
31.    procedure Menu (
32.          My_Menu_Choice :    out Menu_Choice ) is
33.
34.    begin
35.       Ada.Text_Io.Put_Line("_____________________________________________");
36.       Ada.Text_Io.Put_Line("JK's Program to Implement Simple Math Functions");
37.       Ada.Text_Io.Put_Line("_____________________________________________");
38.       Ada.Text_Io.Put_Line("1. Add Two Numbers");
39.       Ada.Text_Io.Put_Line("2. Multiply Two Integers");
40.       Ada.Text_Io.Put_Line("3. Quit");
41.       Ada.Text_Io.Put("Please Enter Your Choice (1-3) : ");
42.       Ada.Integer_Text_Io.Get(My_Menu_Choice);
43.    end Menu;
44.
45.
46.
47. end My_Math_Package;
```

47 lines: No errors

C-10 3.
Algorithm

1. Display the menu to the user
2. Get the menu choice from the user
3. If Choice is 1 then
    a. Prompt the user for two floating point numbers
    b. Clear the screen
    c. Compute the sum using the math package
    d. Display the answer in the required format.
4. If Choice is 2 then
    a. Prompt the user for two integer numbers
    b. Clear the screen
    c. Compute the product using the math package
    d. Display the answer in the required format.
5. If Choice is 3 then
    a. Exit the program

Code Listing

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

Compiling: c:/docume~2/jk/desktop/16070/codeso~1/test_math.adb (source file time stamp: 2003-09-24 03:47:18)

```
 1. ------------------------------------------------------------
 2. -- Program to implement a menu driven program using the
 3. -- the math package
 4. -- Programmer : Jayakanth Srinivasan
 5. -- Date Last Modified : 09/23/2003
 6. ------------------------------------------------------------
 7. with My_Math_Package;
 8. with Ada.Text_Io;
 9. with Ada.Float_Text_Io;
10. with Ada.Integer_Text_Io;
11. with Screen;
12.
13. procedure Test_Math is
14.    Choice : My_Math_Package.Menu_Choice;
15.    X,
16.    Y      : Integer;
17.
18.    Number_X,
19.    Number_Y : Float;
20.
21. begin
22.    loop
23.       -- obtain the choice from the user
24.       My_Math_Package.Menu(Choice);
25.       -- exit if the user chooses 3
```

```
26.     exit when Choice = 3;
27.
28.     case Choice is
29.       when 1 =>
30.         -- obtain two floating point numbers
31.         Ada.Text_Io.Put ("Please Enter the Value of X : ");
32.         Ada.Float_Text_Io.Get(Number_X);
33.         Ada.Text_Io.Skip_Line;
34.
35.         Ada.Text_Io.Put("Please Enter the Value of Y : ");
36.         Ada.Float_Text_Io.Get(Number_Y);
37.         Ada.Text_Io.Skip_Line;
38.
39.         -- clear the screen
40.         Screen.Clearscreen;
41.
42.         -- display the results
43.         Ada.Text_Io.Put("Adding");
44.         Ada.Float_Text_Io.Put(Number_X);
45.         Ada.Text_Io.Put("and");
46.         Ada.Float_Text_Io.Put(Number_Y);
47.         Ada.Text_Io.Put(":");
48.
49.         Ada.Text_Io.New_Line;
50.
51.         Ada.Float_Text_Io.Put(Number_X);
52.         Ada.Text_Io.Put("+");
53.         Ada.Float_Text_Io.Put(Number_Y);
54.         Ada.Text_Io.Put("=");
55.         Ada.Float_Text_Io.Put(My_Math_Package.Add(Number_X, Number_Y));
56.         Ada.Text_Io.New_Line;
57.
58.       when 2=>
59.         -- obtain two integers
60.         Ada.Text_Io.Put ("Please Enter the Value of X : ");
61.         Ada.Integer_Text_Io.Get(X);
62.         Ada.Text_Io.Skip_Line;
63.
64.         Ada.Text_Io.Put("Please Enter the Value of Y : ");
65.         Ada.Integer_Text_Io.Get(Y);
66.         Ada.Text_Io.Skip_Line;
67.         -- clear the screen
68.         Screen.Clearscreen;
69.
70.         -- display the product
71.         Ada.Text_Io.Put("Multiplying");
72.         Ada.Integer_Text_Io.Put(X);
73.         Ada.Text_Io.Put("and");
74.         Ada.Integer_Text_Io.Put(Y);
75.         Ada.Text_Io.Put(":");
76.
77.         Ada.Text_Io.New_Line;
78.
79.         Ada.Integer_Text_Io.Put(X);
80.         Ada.Text_Io.Put("*");
81.         Ada.Integer_Text_Io.Put(Y);
82.         Ada.Text_Io.Put("=");
83.         Ada.Integer_Text_Io.Put(My_Math_Package.Multiply(X, Y));
```

```
84.
85.             Ada.Text_Io.New_Line;
86.
87.          when 3 =>
88.             -- dont have to do anything, exits at the beginning of the loop
89.             null;
90.       end case;
91.    end loop;
92.
93. end Test_Math;
94.
95.
```

95 lines: No errors

T10 SOLUTIONS (WAIT 2)

a) $\frac{T_T}{T} = 1 + \frac{\gamma-1}{2}M^2 = 1 + 0.2(4) = 1.8$

$\therefore$ $\boxed{T_T = 390.6\ K,\ T = T_{atm} = 217\ K}$

$\frac{P_T}{P} = \left[\frac{T_T}{T}\right]^{\gamma/\gamma-1} = \left[1.8\right]^{1.4/.4} = 7.82$

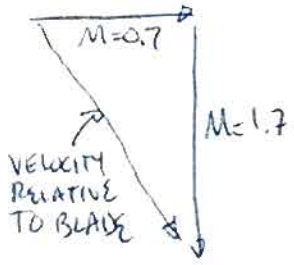
$\therefore$ $\boxed{P_T = 176.8\ kPa,\ P = p_{atm} = 22.6\ kPa}$

b) $\boxed{T_T \text{ \& } P_T \text{ ARE THE SAME AS ABOVE}}$ INLET IS ADIABATIC AND Q-S SO STAG. QUANTITIES ARE CONSTANT AND WE HAVEN'T CHANGED REFERENCE FRAMES)

$T = \frac{T_T}{1 + \frac{\gamma-1}{2}M^2}$, $M = 0.7 \Rightarrow \boxed{T = 355.7\ K}$

$P = \frac{P_T}{\left[1 + \frac{\gamma-1}{2}M^2\right]^{\gamma/\gamma-1}}$, $M = 0.7 \Rightarrow \boxed{P = 127.5\ kPa}$

c) FLOW IN INLET IS $M = 0.7$ (AXIAL) WITH $\boxed{T = 355.7\ K.}$ BLADE MOVING AT $M = 1.7$ (TANGENTIALLY)



VELOCITY RELATIVE TO BLADE = 1.84 Mach

$T_T = T\left(1 + \frac{\gamma-1}{2}M^2\right)$ $T_T = ?$

$T_T = 355.7\left(1 + \frac{\gamma-1}{2}(1.84)^2\right) = \boxed{596\ K}$

$P_T = \boxed{127.5\ kPa}\left(1 + \frac{\gamma-1}{2}(1.84)^2\right)^{\gamma/\gamma-1} = \boxed{779\ kPa}$

$P = ?$ $P_T = ?$

d) $T_{T_{vessel}} = 300K$ ACCELERATE TO $M=2$

$$T_{\substack{test \\ section}} = \frac{T_T}{1 + \frac{\gamma-1}{2} M^2} = \frac{300}{1.8} = \boxed{166.7\,K = T_{test\ section}}$$

FOR $T_{test\ section} = 217K$ (flight temp)

AND MACH = 2, THEN MUST HEAT

VESSEL TO $1.8 \cdot 217K = \boxed{390.6\,K}$

a) APPLY SFEE TO COLD SIDE TO DETERMINE J/s TRANSFERRED AS HEAT, $\dot{Q}$. THEN USE THIS IN SFEE FOR HOT SIDE TO SOLVE FOR $T_{out,\,HOT}$.

SFEE: $q - w_s = h_2 - h_1 + \frac{c_2^2}{2} + \frac{c_1^2}{2}$

$q = \frac{\dot{Q}}{\dot{m}}$ $\quad h_2 - h_1 = c_p (T_2 - T_1)$ (IDEAL GAS)

COLD SIDE:

$$\dot{Q} = 1\ \tfrac{kg}{s}\left[1003.5\ \tfrac{J}{kg\cdot K}\,(350K - 300K) + \frac{(60\ m/s)^2}{2} - \frac{(5\ m/s)^2}{2}\right]$$

$$\boxed{\dot{Q} = 50725\ J/s}$$ (+) SINCE ADDED TO SYSTEM

HOT SIDE: $\dot{Q}_{HOT} = -\dot{Q}_{COLD}$ SINCE REMOVED FROM HOT SIDE

$$-\frac{50725\ J/s}{5\ kg/s} = \left[1003.5\ \tfrac{J}{kg\cdot K}\,(T_{out,\,HOT} - 500K) + \frac{(75\ m/s)^2}{2} - \frac{(100\ m/s)^2}{2}\right]$$

$$\Longrightarrow \boxed{T_{out,\,HOT} = 492\,K}$$

b) NO SHAFT WORK DONE, EVALUATE FLOW WORK

$$w_f = P_{out}v_{out} - P_{in}v_{in} = R(T_{out} - T_{in})$$

$$w_{f_{COLD}} = 287\ \tfrac{J}{kg\cdot K}\,[350K - 300K] = 14.4\ kJ/kg$$

$$\dot{W}_{f_{COLD}} = 14.4\ \tfrac{kJ}{kg}\cdot 1\ kg/s = 14.4\ kW$$ (WORK DONE BY FLOW EXPANDING)

$$\dot{W}_{f_{HOT}} = \frac{5\,kg}{s} \cdot \frac{287\,J}{kg\text{-}K}\left[492\,K - 500\,K\right] = -11.5\,kW$$

(WORK DONE ON FLOW TO COMPRESS IT)

$$\dot{W}_{f_{NET}} = 14.4\,kW - 11.5\,kW$$

$$= 2.87\,kW$$

(SO THERE IS NET WORK DONE BY THE HEAT EXCHANGER, JUST NOT FLOW WORK)

c.) PROCESS IS IRREVERSIBLE — LIKE PUTTING A HOT BRICK NEXT TO A COLD BRICK. (HEAT X-FER ACROSS A FINITE TEMPERATURE DIFFERENCE.) CANNOT PUT SYSTEM BACK TO INITIAL STATE WITHOUT CHANGING THE SURROUNDINGS.

d) THE HOT FLOW ENTERS THE DEVICE WITH HIGH INTERNAL AND KINETIC ENERGY. THE COLD FLOW ENTERS THE DEVICE WITH LOWER KINETIC AND INTERNAL ENERGY. BECAUSE OF THE TEMPERATURE DIFFERENCE, ENERGY FLOWS FROM THE HOT SIDE TO THE COLD SIDE (HEAT TRANSFER), THEREBY RAISING THE KINETIC AND INTERNAL ENERGY OF THE COLD STREAM AND LOWERING THE KINETIC AND INTERNAL OF THE HOT STREAM. THERE IS ALSO FLOW OF ENERGY OUT OF THE SYSTEM DUE TO BOTH OF THE STREAMS EXITING THE DEVICE (INTERNAL ENERGY) AND DUE TO THE NET FLOW WORK DONE ON THE SURROUNDINGS BY THE FLOW ENTERING & LEAVING THE DEVICE.

T12 SOLUTIONS (WAITE)

a) $S_2 - S_1 = C_p \ln\left(\frac{T_2}{T_1}\right) - R \ln\left(\frac{P_2}{P_1}\right)$

q-S, ADIABATIC COMPRESSION

$S_2 - S_1 = 0$

$T_2 = 475$ $P_2 = 500 kPa$
$T_1 = 300$ $P_1 = 100 kPa$

INSTANTANEOUS COMPRESSION:

$S_2 - S_1 = 303 \, J/kg\text{-}K$

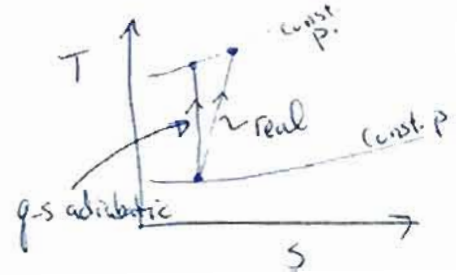
$T_2 = 643.2K$ $P_2 = 500 kPa$
$T_1 = 300K$ $P_1 = 100 kPa$

THE INSTANTANEOUS, IRREVERSIBLE PROCESS REQUIRES MORE WORK TO COMPRESS THE GAS AND RESULTS IN A GREATER CHANGE IN ENTROPY. THE q-S ADIABATIC PROCESS IS REVERSIBLE ($\Delta S = 0$)

b) q-S, ADIABATIC PROCESS: $P_2 = 1473 kPA$, $P_1 = 101.3 kPa$
$T_2 = 638K$, $T_1 = 297K$
$\Delta S = 0$

REAL PROCESS: $P_2 = 1473 kPa$, $P_1 = 101.3 kPa$
$T_2 = 661K$, $T_1 = 297K$

$\Delta S = 1003.5 \ln\left(\frac{661}{297}\right) - 287 \ln(14.54) = 134.6 \, J/kg\text{-}K$



Energy can take many different forms (kinetic energy, internal energy, potential energy, chemical energy). Energy can be exchanged from one form to another (e.g. using fuel to get power out of an engine, or using an electric stove to heat water, or dropping a rock from your hand to the ground) but the only way to change the total energy (the sum of all the various forms of energy) of a system is to add or remove heat or for the system to do work or have work done on it. If heat is added to, or if work is done on a system its energy increases. This is a statement of the First Law of Thermodynamics. It is a very good law but it allows many processes that are impossible in our world.

If I touch a hot and a cold brick together I get two medium temperature bricks. The First Law of Thermodynamics also allows that the opposite can happen. It allows that two medium temperature bricks in contact with each other can spontaneously produce a hot brick and a cold brick, as long as the total energy of the two bricks together hasn't changed. This is impossible in our world, as are many other "reverse" processes: the spontaneous un-mixing of two gases in a container (the reverse of perfume spreading across a room), low pressure air spontaneously collecting itself into a small high pressure volume (the reverse of a balloon breaking), a rock collecting heat from the ground and spontaneously converting it to kinetic and then potential energy to jump into your hand (the reverse of dropping a rock from your hand). Of course, it is possible to reverse all of these processes but only by using work (they won't spontaneously happen). So when we say something is irreversible, we mean that it can not be reversed *without the application of work from the surroundings.*

All real processes are to some degree irreversible. Things that cause processes to be irreversible are unrestrained expansion, friction, molecular diffusion, heat transfer across a finite temperature difference, etc. The Second Law of Thermodynamics provides the tool for sorting out which processes allowed by the First Law are possible and which are not. If the process is possible, a thermodynamic property called entropy will increase (if calculated and summed for both the system and the surroundings). The Second Law also allows us to characterize the "degree of irreversibility" by measuring how much the entropy of the system and surroundings increases as a result of a process. Some processes have more irreversibility (e.g. more friction) than others. The more irreversibilities in a device, the less efficient it is. Reversible processes are not possible, but they are useful idealizations that let us understand the best that can be achieved (we know that a pendulum without friction will run forever, but how efficient could an engine be without friction?). For example, if the power plant at MIT were ideal (no friction, no heat transfer across finite temperature differences, no free expansions, etc.), then all the processes that make it work could run equally well forward and reverse without application of additional work from the surroundings. In this ideal case, application of the First Law of Thermodynamics can be used to show that for each unit of energy that goes into the power plant as fuel, about half that unit of energy comes out as useful work. However, for the real MIT power plant , with all its irreversibilities, each unit of fuel energy only results in about 1/4 of a unit of useful work. Quite a significant impact indeed.

Problem S1 Solutions

1.a

$$x + y - 2z = -1$$
$$x + 4y + 2z = 5$$
$$x + y - z = 0$$

$$x + y - 2z = -1 \qquad x + 4y + 2z = 5$$
$$x + 4y + 2z = 5 \qquad 2(x + y - z) = 0$$
$$2x + 5y = 4 \qquad 3x + 6y = 5$$

$$-2(3x + 6y = 5)$$
$$3(2x + 5y = 4)$$
$$3y = 2$$
$$\boxed{y = 2/3}$$
$$3x + 6(2/3) = 5$$
$$3x = 1$$
$$\boxed{x = 1/3}$$
$$1/2 + 2/3 - z = 0$$
$$\boxed{z = 1}$$

1.b

$$\left[\begin{array}{ccc|c} 1 & 1 & -2 & -1 \\ 1 & 4 & 2 & 5 \\ 1 & 1 & -1 & 0 \end{array}\right] \text{. Subtract row 1 from row 2}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & -2 & -1 \\ 0 & 3 & 4 & 6 \\ 1 & 1 & -1 & 0 \end{array}\right] \text{. Subtract row 1 from row 3}$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 1 & -2 & -1 \\ 0 & 3 & 4 & 6 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

$$\boxed{z = 1}$$
$$\boxed{y = 2/3}$$
$$\boxed{x = 1/3}$$

1.c

$$x = \frac{\begin{vmatrix} -1 & 5 & -2 \\ 5 & 4 & 2 \\ 0 & 1 & -1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -2 \\ 1 & 4 & 2 \\ 1 & 1 & -1 \end{vmatrix}} = \frac{1}{3} \Rightarrow \boxed{x = 1/3}$$

$$y = \frac{\begin{vmatrix} 1 & -1 & -2 \\ 1 & 5 & 2 \\ 1 & 0 & -1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -2 \\ 1 & 4 & 2 \\ 1 & 1 & -1 \end{vmatrix}} = \frac{2}{3} \Rightarrow \boxed{y = 2/3}$$

$$z = \frac{\begin{vmatrix} 1 & 1 & -1 \\ 1 & 4 & 5 \\ 1 & 1 & 0 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -2 \\ 1 & 4 & 2 \\ 1 & 1 & -1 \end{vmatrix}} = \frac{3}{3} \Rightarrow \boxed{z = 1}$$

$$\det\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = a(ei - fh) - b(di - fg) + c(dh - eg)$$

2.a

$$4x + 2y + 2z = 7$$
$$3x + y + 2z = 5$$
$$x + 3y - z = 4$$

$$4x + 2y + 2z = 7 \qquad 3x + y + 2z = 5$$
$$\underline{3x + y + 2z = 5} \qquad \underline{2(x + 3y - z = 4)}$$
$$x + y = 2 \qquad 5x + 7y = 13$$

$$-5(x + y = 2)$$
$$\underline{5x + 7y = 13}$$
$$2y = 3$$

$$\boxed{y = 3/2}$$
$$x + 3/2 = 2$$
$$\boxed{x = 1/2}$$
$$1/2 + 3(3/2) - z = 4$$
$$\boxed{z = 1}$$

2.b

$$\left[\begin{array}{ccc|c} 4 & 2 & 2 & 7 \\ \underline{3} & 1 & 2 & 5 \\ \underline{1} & 3 & -1 & 4 \end{array}\right] =$$
. Multiply row 1 by -3/4 and add to row 2

$$\Rightarrow = \left[\begin{array}{ccc|c} 4 & 2 & 2 & 7 \\ \underline{0} & -1/2 & 1/2 & -1/4 \\ \underline{1} & 3 & -1 & 4 \end{array}\right] =$$
. Multiply row 1 by -1/4 and add to row 3

$$\Rightarrow = \left[\begin{array}{ccc|c} 4 & 2 & 2 & 7 \\ \underline{0} & -1/2 & 1/2 & -1/4 \\ \underline{0} & 5/2 & -3/2 & 9/4 \end{array}\right] =$$
. Multiply row 2 by 5 and add to row 3

$$\Rightarrow = \left[\begin{array}{ccc|c} 4 & 2 & 2 & 7 \\ \underline{0} & -1/2 & 1/2 & -1/4 \\ \underline{0} & 0 & 1 & 1 \end{array}\right] =$$

$$\boxed{z = 1}$$
$$\boxed{y = 3/2}$$
$$\boxed{x = 1/2}$$

2.c

$$x = \frac{\begin{vmatrix} -1 & 1 & -2 \\ 5 & 4 & 2 \\ 0 & 1 & -1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -2 \\ 1 & 4 & 2 \\ 1 & 1 & -2 \end{vmatrix}} \quad \frac{1}{3} \Rightarrow x = 1/3$$

$$y = \frac{\begin{vmatrix} 1 & -1 & -2 \\ 1 & 5 & 2 \\ 1 & 0 & -1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -2 \\ 1 & 4 & 2 \\ 1 & 1 & -2 \end{vmatrix}} \quad \frac{2}{3} \Rightarrow y = 2/3$$

$$z = \frac{\begin{vmatrix} 1 & 1 & -1 \\ 1 & 4 & 5 \\ 1 & 1 & 0 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -2 \\ 1 & 4 & 2 \\ 1 & 1 & -2 \end{vmatrix}} \quad \frac{3}{3} \Rightarrow z = 1$$