For all problems in problem set 3:

Solutions to **Part 3** must be submitted both electronically (on server \\aero-astro\16.35) and on paper.

Problem sets should have a title in the following format: LastName_PSXx.ad[bs]. Due date: **10/23/02, 3pm.**

2) All submitted problems must contain your name, email address, and which problem you are submitting.
3) The following should be written as comments in the code: a short description of what the program is doing, and also document what the most important variables, data structures, and constants are, and how they are used in the program.
4) In addition to 3) each module should have a "header comment" with the following information:

```
------------------
-- Module name: Name of module
-- Explanation: What is the purpose of this module. Explain it so a person
--              unfamiliar with the code can understand.
-- Input: Input to this module from the calling unit.
-- Output: Data that is returned to calling unit (for procedures).
-- Return value: Value that is returned to calling unit (for functions).
-- Comments: Extra information that might be needed for the unit to execute properly.
```

Generics and Tasking

Problem 1 - Set_Of

Write the body for the generic package Set_Of that enables the manipulation of sets of an arbitrary type. The specification of the package looks like:

```
generic
  type Element is (<>);
package Set_Of is
  type Set is private;
  type List is array (Positive range <>) of Element;

  Empty, Full: constant Set;

  function Make_Set(L: List) return Set;
  function Make_Set(E: Element) return Set;
  function Decompose(S: Set) return List;

  function "+" (S, T:Set) return Set;  --union
  function "*" (S, T:Set) return Set;  --intersection
  function "-" (S, T:Set) return Set;  --symmetric difference
  function "<" (E: Element; S:Set) return Boolean; --inclusion
  function "<=" (S, T: Set) return Boolean;   --contains
  function Size(S: Set) return Natural;       --no of elements

  private
  type Set is array (Element) of Boolean;

  Empty: constant Set := (Set'range => False);
  Full:  constant Set := (Set'range => True);
end;
```

The single generic parameter is the element type which must be discrete. The type `Set` is made private so that the Boolean operations cannot be directly applied. Aggregates of the type `List` are used to represent literal sets. The constants `Empty` and `Full` denote the empty and full set respectively. The function `Make_Set` enable the creation of a set from a list of the element values or a single element value. `Decompose` turns a set back into a list of elements.
In the private part the type `Set` is declared as a Boolean array indexed by the element type (which is why the element type had to be discrete). The constants Empty and Full are declared as arrays whose elements are all False and all True respectively.

Problem 2 – Set_Of again

Rewrite the private part of `Set_Of` so that an object of the type `Set` is by default given the initial value `Empty` when declared.

Problem 3 - Rendezvous

Note: this is simply an exercise on using the rendezvous. A better solution is to use a protected object.
Write the body of a task whose specification is:

```
task Char_To_Line is
   entry Put(C: in Character);
   entry Get(L: out Line);
end;
```

where

```
type Line is array (1 .. 80) of Character;
```

the task acts as a buffer which alternately builds up a line by accepting successive calls of `Put` and then delivers a complete line on a call of `Get`.

Problem 4 – Protected Object

Reconsider Problem 3 above using a protected object.