**Ada Fundamentals with GNAT: Workshop 2**

1. Enumeration “math”

Objective: This problem illustrates enumeration types and program structure.

The workshop directory contains a main procedure Day\_Test and the following package specification

**package** Day\_Pkg **is**  
 **type** Day **is** (Sun, Mon, Tue, Wed, Thu, Fri, Sat);  
 **function** "+"(D : Day; N : Integer) **return** Day;  
 **function** "-"(D1, D2 : Day) **return** Integer);  
**end** Day\_Pkg;

The operation D+N should return a Day value corresponding to N days after D. For example. Mon + 3 = Thu, Fri + 14 = Fri, Sun + (-2) = Fri.

Analogously, D1-D2 is the smallest non-negative number N such that D2+N = D1 (*i.e.*, the number of days from D2 to D1.) Thus Fri-Mon = 4; Mon-Sat = 2.

Implement the package body and test it with the main procedure.

2. Vector package

Objective: This problem illustrates array features and program structure

The workshop directory contains source files for Vector\_Pkg (specification and partially implemented body) and a main procedure Vector\_Test, based on the class notes (slides 24 and 25 in Part 2, page 2/13 in the course notebook). Complete the implementation of the package body.

3. Random permutations

Objective: This problem illustrates array features

The workshop directory contains a main procedure Array\_Test and the following package specification:

**package** Array\_Handling **is**  
 **type** Array\_Type **is array**( Positive **range** <> ) **of** Float;  
 **procedure** Shuffle( Data : **in out** Array\_Type );  
 *-- Rearranges Data through a random permutation***end** Array\_Handling;

Implement the package body. Reuse the Random\_Numbers package from Workshop 1.

Note: generating a random permutation is not completely trivial. Check with the instructor if you need some guidance.

4. Recovery from data input errors

Objective: This problem illustrates exception handling

The procedure Day\_Test from Exercise 1 is not very robust; if the user enters improperly formed input at any of the prompts, the program terminates with an unhandled exception. Modify the program as follows:

* if an incorrect Day value is entered, the program will substitute the value Sun and output a message to this effect.
* if an incorrect integer is entered, the program will iterate the request for input until a proper value is received

5. Propagating and handling program-defined exceptions

Objective: This problem illustrates exception handling

The package Vector\_Pkg\_2 adds the following declarations to Vector\_Pkg from Exercise 2:

**function** "\*"(L, R: Vector) **return** Float;  
Parameter\_Length\_Error : **exception**;

Implement the “\*” function to return the “dot product” of L and R (i.e. the sum of the products of corresponding elements of L and R), where L'Length=R'Length is the required precondition. If the lengths of the parameters are not the same, then the function raises Parameter\_Length\_Error.

The procedure Vector\_Test\_2 tests the function but does not handle the Parameter\_Length\_Error exception. Modify the procedure to handle the exception by displaying a message, without terminating the loop. The message displayed should be the string returned from the function Ada.Exceptions.Exception\_Information.

6. Alternative style to Integer'Value

Objective: This problem illustrates exception handling

The predefined attribute function Integer'Value converts a String to an Integer, raising an exception if the String does not spell out a valid Integer value. It is sometimes more convenient to return a status value instead of raising an exception. Implement the following procedure to have such an effect:

**procedure** Integer\_Value(Item : **in** String;  
 Result : **out** Integer;  
 Valid : **out** Boolean);

If Integer'Value(Item) does not raise an exception, then Result should be the integer returned from this invocation, and Valid should be True. Otherwise Valid should be False, and Result should not be assigned a value. Use the main program in the workshop directory to test your implementation.