SWEN90010: HIGH INTEGRITY SYSTEMS ENGINEERING
DEPARTMENT OF COMPUTING AND INFORMATION SYSTEMS
THE UNIVERSITY OF MELBOURNE

Workshop Pointers in SPARK

Introduction

In the lectures, we discussed the difficulties of reasoning about programs with pointers, and how SPARK tackles this problem by limiting unsafe aliasing. In this workshop you will experiment with the SPARK Prover to see how these ideas work in the context of a tiny example.

Exercises

The focus of this workshop is a SPARK package that defines a record data type `Person` to store information about an individual: their `Name` and `Age` (in years) respectively.

Because the name string could be of arbitrary length, rather than storing the string directly inside the `Person` record, we instead store a *pointer* to the name string.

We additionally use Ada's type system to check that the pointer is never null.

```
type Years is range 0..150;

type NamePtr is not null access String;

type Person is record
   Name : NamePtr;
   Age : Years;
end record;
```

The package includes some procedures and the example includes a tiny main driver program, `main.adb`.

1. Open the project, using the provided `default.gpr` project file. Compile the source code the GPS environment, read the code and run it to see what it does.

2. Now run the SPARK prover: *SPARK* → *Prove All*. You will see that the provided `main.adb` gives an error on its very first line that does `Ada.Text_IO.Put_Line(S.all)`.

   Why does this error occur? What potential problem is the error telling us about with this code?

3. To fix this problem, comment out the problematic line. Then run the SPARK Prover again. You should now see a warning message about the `Get_Name` procedure of the `Person` package.

   What is this warning about?

   Hint: remember that SPARK enforces the rule that when aliases exist to the same data in memory, that we are only allowed to use the alias that was most recently created.

   Hint: why is this procedure problematic but `Swap_Name` is not?

4. How can we fix this problem? Modify the `Get_Name` procedure to fix the problem. You might need to adjust the postcondition annotation on the procedure afterwards. Then re-run the SPARK Prover to check that your modification is OK.

5. Look at the call to `Name_Compare` function in `main.adb`. Notice how we explicitly pass it two references that alias!

   Why is this allowed by the SPARK Prover?

   To understand, add a line of code in `Name_Compare` that modifies one of the strings and re-run the SPARK Prover again.

   Hint: recall that in SPARK, functions must be free of side-effects. This is why this function is not allowed to modify its arguments. But it also has implications for what kinds of aliasing are allowed, compared to code that might have side-effects.