Name: Luka Aitken

Student ID: T00663672

Date: March 26, 2025

SENG 4630 – Lab 10 – Generics

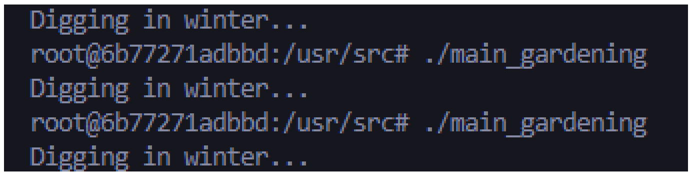
Task 1:

```
-- Main_gardening.adb
-- Luka Aitken

--T00663672
--Task 1
with Ada.Text_IO; use Ada.Text_IO;
with Gardening;

procedure Main_Gardening is
   procedure Print_Task(T : String) is
   begin
      Put_Line(T);
   end Print_Task;

   package My_Gardening is new Gardening (Print_Task);
begin
   My_Gardening.Do_Gardening(My_Gardening.Month_Type'Val(0));
end Main_Gardening;
```

```
-- gardening.adb
-- Luka Aitken
--T00663672
--Task 1
with Ada.Text_IO; use Ada.Text_IO;
package body Gardening is
   procedure Do_Gardening(M : Month_Type) is
   begin
      case M is
         when December | January | February =>
            Perform_Task("Dig");
         when March | April | May =>
            Perform_Task("Sow");
         when June | July | August =>
            Perform_Task("Tend");
         when September | October | November =>
            Perform_Task("Harvest");
      end case;
   end Do_Gardening;
end Gardening;
```

```
-- gardening.ads
-- Luka Aitken
--T00663672
--Task 1
generic
   with procedure Perform_Task(T : String);
package Gardening is
   type Month_Type is (January, February, March, April, May, June,
                       July, August, September, October, November, December);

   procedure Do_Gardening(M : Month_Type);
end Gardening;
```

Output for task 1:



```
Digging in winter...
root@6b77271adbbd:/usr/src# ./main_gardening
Digging in winter...
root@6b77271adbbd:/usr/src# ./main_gardening
Digging in winter...
```

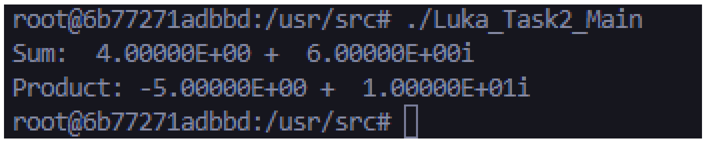
Task 2:

```
-- Luka_Task2_Main.adb
-- Luka Aitken
--T00663672
--Task 2
with Ada.Text_IO; use Ada.Text_IO;
with Luka_Task2; use Luka_Task2;

procedure Luka_Task2_Main is
    C1, C2, C3 : Complex;
begin
    C1.Real_Part := 3.0;
    C1.Imag_Part := 4.0;
    C2.Real_Part := 1.0;
    C2.Imag_Part := 2.0;

    C3 := Add(C1, C2);
    Put_Line("Sum: " & Float'Image(C3.Real_Part) & " + " &
Float'Image(C3.Imag_Part) & "i");

    C3 := Multiply(C1, C2);
    Put_Line("Product: " & Float'Image(C3.Real_Part) & " + " &
Float'Image(C3.Imag_Part) & "i");
end Luka_Task2_Main;
```

```
-- Luka_Task2.adb
-- Luka Aitken
--T00663672
--Task 2
package body Luka_Task2 is
    function Add (C1, C2 : Complex) return Complex is
    begin
        return (Complex'(
            Real_Part  => C1.Real_Part + C2.Real_Part,
            Imag_Part  => C1.Imag_Part + C2.Imag_Part
        ));
    end Add;
    function Multiply (C1, C2 : Complex) return Complex is
    begin
        return (Complex'(
            Real_Part  => (C1.Real_Part * C2.Real_Part) - (C1.Imag_Part *
C2.Imag_Part),
            Imag_Part  => (C1.Real_Part * C2.Imag_Part) + (C1.Imag_Part *
C2.Real_Part)
        ));
    end Multiply;

end Luka_Task2;
```

```
-- Luka_Task2.ads
-- Luka Aitken
--T00663672
--Task 2
package Luka_Task2 is
    type Complex is record
        Real_Part  : Float;
        Imag_Part  : Float;
    end record;
    function Add (C1, C2 : Complex) return Complex;
    function Multiply (C1, C2 : Complex) return Complex;
end Luka_Task2;
```

Output for Task2:



```
root@6b77271adbbd:/usr/src# ./Luka_Task2_Main
Sum:  4.00000E+00 +  6.00000E+00i
Product: -5.00000E+00 +  1.00000E+01i
root@6b77271adbbd:/usr/src# 
```

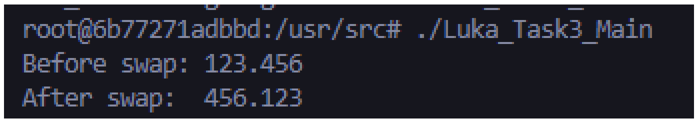
Task 3:

```ada
-- Luka_Task3_Main.adb
-- Luka Aitken
--T00663672
--Task 3
with Ada.Text_IO; use Ada.Text_IO;
with Luka_Task3;
with Ada.Float_Text_IO;

procedure Luka_Task3_Main is
   procedure Swap is new Luka_Task3.Swap_Float_Parts (Float);

   Test_Value : Float := 123.456;
begin
   Put("Before swap: ");
   Ada.Float_Text_IO.Put(Test_Value, Fore => 1, Aft => 3, Exp => 0);
   New_Line;

   Swap(Test_Value);

   Put("After swap:  ");
   Ada.Float_Text_IO.Put(Test_Value, Fore => 1, Aft => 3, Exp => 0);
   New_Line;
end Luka_Task3_Main;
```

```ada
-- Luka_Task3.adb
-- Luka Aitken
--T00663672
--Task 3
package body Luka_Task3 is
   procedure Swap_Float_Parts (X : in out Float_Type) is
      Whole_Part : Float_Type;
      Frac_Part  : Float_Type;
      Temp       : Float_Type;
   begin
      Whole_Part := Float_Type'Truncation(X);

      Frac_Part := X - Whole_Part;

      Temp := Frac_Part * 1000.0;
      X := Float_Type'Truncation(Temp) +
           (Whole_Part / 1000.0);
   end Swap_Float_Parts;
end Luka_Task3;
```

```
-- Luka_Task3.ads
-- Luka Aitken
--T00663672
--Task 3
package Luka_Task3 is
   generic
      type Float_Type is digits <>;
   procedure Swap_Float_Parts (X : in out Float_Type);
end Luka_Task3;
```

Output for task3:



```
root@6b77271adbbd:/usr/src# ./Luka_Task3_Main
Before swap: 123.456
After swap:  456.123
```