Aufgabe 3.6

1. Fehler, smart_pointers.adb, Zeile 18, `function Is_Null`:

```
return X.Reference_Count = 0;
```

Änderung

```
return X=null or else X.Reference_Count = 0;
```

Testfall:

```
Assert (Is_Null (Null_Pointer));
```

Ausgabe (ohne Korrektur):

```
raised CONSTRAINT_ERROR : smart_pointers.adb:18 access check
failed
```

Ausgabe (mit Korrektur):

```
Good
```

Begründung:
Die Funktion `Is_Null` versuchte einen nichtexistierenden Objekt zuzugreifen.

2. Fehler, smart_pointers.adb, Zeile 8, `procedure Create`:

```
X := new Info'(0, Datums.Allocate);
```

Änderung:

```
X := new Info'(1, Datums.Allocate);
```

Testfall:

```
Smart_Pointers.Create (A);
Assert (not Is_Null (A));
```

Ausgabe (ohne Korrektur):

```
Allocate object 1
FAILED
```

Ausgabe (mit Korrektur):

```
Allocate object 1
Good
```

Begründung:
Ein Objekt wurde mit 0 Referenzen darauf erzeugt und gelte deshalb als Null, obwohl es einen Smart Pointer auf dieses Objekt gab.

3. Fehler, smart_pointers.adb, Zeile 52, `procedure Release:`

```
Dec (X);
```

Änderung:

```
(Zeile löschen)
```

Testfall:

```
Smart_Pointers.Create (A);
Smart_Pointers.Assign (A, B);
Smart_Pointers.Release (B);
Smart_Pointers.Print (A);
```

Ausgabe (ohne Korrektur):

```
Allocate object 1
Free object 1
ERROR: Printing deallocated object 1
```

Ausgabe (mit Korrektur):

```
Allocate object 1
Printing object 1
```

Begründung:

Es gab doppelte Dekrementierung (einmal in `procedure Release` und einmal in `procedure Assign`), deshalb wurde das Datum gelöscht.