```
(*
                              CS51 Lab 6
     Variants, algebraic types, and pattern matching (continued)

In this lab, you will continue to use algebraic data types to create
several data structures. You'll create a data structure for storing
weather information and work with a specific type of binary tree, the
*binary search tree*, which allows for efficient storage and search of
ordered information. A particular application is the use of Gorn
addresses, named after the early computer pioneer Saul Gorn of
University of Pennsylvania, who invented the technique.

There are two parts to this lab. Please refer to the following
files to complete all exercises:

-> lab6_part1.ml -- Part 1: The Weather
   lab6_part2.ml -- Part 2: Binary search trees and Gorn addresses
 *)

(* Objective: This lab is intended to reinforce core concepts in
   typing in OCaml, including:

     Algebraic data types
     Using algebraic data types to enforce invariants
     Implementing polymorphic algebraic data types
 *)

(*======================================================================
                         Part 1: The Weather
                   Variants and invariants revisited

As the saying goes, everybody talks about the weather, but nobody
*does* anything about it. Let's remedy that. We'll implement a data
structue for storing seasonal weather information.

......................................................................
Exercise 1: Define a season type

There are four seasons -- spring, summer, autumn, and winter. Define an
enumerated type that allows for just these four values.
......................................................................*)

type season = ToBeDefined

(*......................................................................
Exercise 2: Define a type providing the weather condition

For our purposes, there are three weather conditions -- sunny, rainy,
and snowing. Each condition (except for "sunny", of course) has an
associated precipitation amount, an integer specifying the
precipitation in millimeters.

Define a type providing the possible weather conditions. Remember the
invariant: no precipitation amount for sunny conditions.
......................................................................*)

type condition = ToBeDefined

(*......................................................................
Exercise 3: Define a weather status type combining season and condition

The `weather_status` type should combine a season together with a
condition. Use a record type for the combination, with appropriate
field names.
......................................................................*)
```

```
type weather_status = ToBeDefined

(* For consistency with our unit testing, we should make sure that the
types you defined match with the types we're testing against. You
shuld call over a staff member to verify the types you've
specified. Alternatively, we've placed our expected data types at
url.cs51.io/lab6types *)

(*.....................................................................
Exercise 4: For each of the following cases, construct a value of type
`weather_status` that represents the case. If no such value exists,
specify a value representing a sunny day in summer.

Case 1: a sunny day in winter with 2 mm of snow
Case 2: a fall day with 5 mm of rain
Case 3: a snowy winter day with 100 mm of snow
.....................................................................*)

let case1 = failwith "case 1 not implemented" ;;
let case2 = failwith "case 2 not implemented" ;;
let case3 = failwith "case 3 not implemented" ;;

(* Now for some functions that manipulate these weather values.

.....................................................................
Exercise 5: Calculating precipitation

Define a function `precipitation_amount : condition -> int` that given
a weather condition, returns the precipitation amount (0 mm for sunny
days of course).
.....................................................................*)

let precipitation_amount (condition : condition) : int =
  failwith "precipitation_amount not implemented"

(*.....................................................................
Exercise 6: Strings for a season

Define a function `season_to_string : season -> string` that given a
season, returns a string naming that season. This may come in handy in
Exercise 7.
.....................................................................*)

let season_to_string (season : season) : string =
  failwith "season_to_string"

(*.....................................................................
Exercise 7: Describing the weather

Define a function `describe_weather : weather_status -> string` that
given a weather status, returns a string describing the weather. Here
are some example strings that the function should return given
appropriate inputs.

    "It's snowing in winter. Precipitation: 20 mm."
    "It's a sunny summer day."
    "It's raining in autumn. Precipitation: 100 mm."
.....................................................................*)

let describe_weather (status : weather_status) : string =
  failwith "describe_weather not implemented"
```