OCaml Beginner Exercises

Exercise 1: Basic Functions

Write a function `double` that takes an integer and returns twice its value.

```
(* Test *)
assert (double 4 = 8)
assert (double (-3) = -6)
```

Exercise 2: Pattern Matching

Create a function `describe_number` that takes an integer and returns:

- "positive" if > 0
- "negative" if < 0
- "zero" if = 0

```


(* Test *)
assert (describe_number 5 = "positive")
assert (describe_number (-2) = "negative")
assert (describe_number 0 = "zero")
```

Exercise 3: Lists

Write three functions:

1. `sum_list`: Calculate sum of a list
2. `max_list`: Find maximum value (assume non-empty list)
3. `is_sorted`: Check if list is sorted in ascending order

```
(* Tests *)
assert (sum_list [1;2;3;4] = 10)
assert (max_list [1;3;2;5;4] = 5)
assert (is_sorted [1;2;3;4] = true)
assert (is_sorted [1;3;2;4] = false)
```

Exercise 4: Custom Types

Define a type `shape` that can be Circle, Rectangle, or Triangle. Write a function to calculate area.

```

```

```
(* Tests *)
assert (abs_float(area (Circle 2.0) -. 12.56636) < 0.0001)
assert (area (Rectangle (2.0, 3.0)) = 6.0)
assert (area (Triangle (4.0, 3.0)) = 6.0)
```

Exercise 5: Higher-Order Functions

Implement:

1. `map_option`: Maps a function over an option type
2. `compose`: Function composition

```
(* Tests *)
assert (map_option double (Some 3) = Some 6)
assert (map_option double None = None)

let add_one x = x + 1
let times_two x = x * 2
assert (compose add_one times_two 3 = 7)
```

Each exercise introduces key OCaml concepts:

1. Basic function definition and arithmetic
2. Pattern matching and guards
3. Recursive functions and list processing
4. Algebraic data types
5. Higher-order functions and options

Test your solutions in OCaml REPL (utop) or by creating a .ml file.