CISS445: Programming Languages
Assignment 3

OBJECTIVES

This is the first of several OCAML assignments where the main object is to help you learn the basic OCAML language, including basic types and operators, tuples and lists with their operations, declarations, recursion, and matchings.

This is an easy assignment. Therefore no discussion is allowed.

You MUST refer to a02 for general assignment instructions.

Q1. Write a function power so that `(power x n)` returns the power of `x` to the `n`-th power. Assume that `x` is an integer. Ignore the case where `n` is < 0. You should use the "obvious" recursion, i.e.

$$x^n = \begin{cases} xx^{n-1} & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

Tests	Expected value
`power 2 0`	`1`
`power 0 5`	`0`
`power 0 1`	`0`
`power 2 2`	`4`
`power (-3) 3`	`-27`

Q2. Write a function `power2` that has the same output as power from Q1 but is computed using this recursion:

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x^{n/2}x^{n/2} & \text{if } n > 0 \text{ is even} \\ xx^{(n-1)/2}x^{(n-1)/2} & \text{if } n > 0 \text{ is odd} \end{cases}$$

Q3. Write a function `sum_to` such that `(sum_to n)` computes the sum from `0` to `n`. If `n` is negative, then `0` is returned.

Tests	Expected value
sum_to 0	0
sum_to (-1)	0
sum_to 1	1
sum_to 2	3
sum_to 3	6
sum_to 4	10

Q4. Write a function `isprime` such that `(isprime n)` is `true` if and only if `n` is a prime.

Tests	Expected value
isprime 0	false
isprime 1	false
isprime 2	true
isprime 3	true
isprime 4	false
isprime 5	true
isprime 6	false
isprime 7	true
isprime 8	false
isprime 9	false
isprime 10	false
isprime 11	true

Q5. Write a function `head` that returns the first element of a list. (Ignore the case where the list passed in is empty).

Tests	Expected value
`head [1]`	`1`
`head [2; 1]`	`2`
`head [3; 2; 1]`	`3`
`head [1.1, 2.2]`	`1.1`

Q6. Write a function `tail` that returns the elements of a list (as a list) without the head. If the list is empty, the empty list is returned. Use pattern matching.

Tests	Expected value
`tail []`	`[]`
`tail [1]`	`[]`
`tail [3; 1]`	`[1]`
`tail [4; 5; 1]`	`[5;1]`

Q7. Write a function `second` that returns the second element of a list. (Hint: Use what you have already written. Ignore the case where the list does not have a second element)

Tests	Expected value
`second [1; 2]`	2
`second [3; 1; 2]`	1
`second [4; 3; 2; 1]`	3
`second [3.3; 4.4; 5.5]`	4.4