

Lab - 2

Week of Jan. 14, 2018

Questions 1-3 should be tried in the OCaml toplevel.

1. Set the variables `x`, `y` and `z` to any *positive* integers of your choice and compute the following expressions:

- (a) $x^3 + 2xyz^2yz + 1$
- (b) Cube root of xyz
- (c) $\ln(x + \sqrt{x^2 + 1})$
- (d) The sine function in OCaml takes in the angle in radians. However, we would like to supply the angle in degrees. Therefore, there is a need to convert the angle in degrees to the angle in radians. Write the expression to do so. Use the following value of π discussed in class: $4\text{atan}(1)$ and the formula for converting degrees into radians as: $\text{radians} = \text{degrees} * \frac{\pi}{180}$.

Use the following web page (the section on floating point arithmetic) for help: <https://caml.inria.fr/pub/docs/manual-ocaml/libref/Pervasives.html>

2. Write the following functions for each of the above:
 - (a) `poly` (takes in 3 parameters, returns one value)
 - (b) `mcuberoot` (takes in 3 parameters, returns one value)
 - (c) `nlog` (takes in 1 parameter, returns 1 value)
 - (d) `degrees_to_radians` (takes in 1 parameter, returns 1 value)
3. We would like to ensure that the inputs to the functions above are correctly bound. Write input validation (or correction) functions for each of the above:
 - (a) `check_poly`: takes in 3 parameters provided as input, returns true only if each parameter is a positive integer. If any of the parameters is negative, then it returns false.
 - (b) `check_mcuberoot`: takes in 3 parameters, returns true only if xyz is positive, otherwise, returns false

- (c) `check_nlog`: takes in 1 parameter, determine for yourself when this function should return true.
- (d) `check_degrees_to_radians`: takes in 1 parameter, returns a value (in degrees) that is in between 0 and 360. NOTE: If the parameter is 361 degrees, then this function returns 1 degree, if the value is -1 degree, then it returns 359 degrees, etc.

4. Source files

- (a) Write function `poly` and the corresponding input validation function in the file `poly.ml`.
- (b) Write a main function that calls function `poly` with the following values: $x = 2, y = 3, z = 4$ and prints out the value returned by the function.
- (c) Compile `poly.ml` into the executable `poly`
- (d) Run `poly`. Ensure that the behaviour of the program is as expected.

5. Remove the main function from the `poly.ml`

- (a) Load the functions in `poly.ml` into the toplevel.
- (b) Test the functionality with the following sets of values:

$$x = -5, y = 3, z = 8$$

$$x = 2, y = 1.2, z = 0$$

$$x = 5, y = -3, z = 2$$

- (c) Think of more test cases. What errors can occur?

6. Repeat steps 4 and 5 for the other functions in 2). You can choose whatever file name you like.

7. Prepare and submit the following files for submission:

- (a) `poly.ml` (which implements 1a and 2a)
- (b) `mcuberoot.ml` (which implements 1b and 2b)
- (c) `nlog.ml` (which implements 1c and 2c)
- (d) `degrees_to_radians.ml` (which implements 1d and 2d)