

Taylor Expansion Again (60)

1. Start by researching how to write comments in OCaml. Then explore the exponential function `**` in OCaml. What is its type signature? Write your result of this question as a comment below.

Hint: In the toplevel, you can type `(**);;` to determine this. Note the added space before `**` ensures it isn't interpreted as a comment.

2. Implement a factorial function `factorial` with a type signature `int -> int`. For example, computing the factorial of 5 should yield a result of 120. Fill in the following:

```
let rec factorial n =  
    (*TODO*)
```

3. Design a Taylor expansion function `taylor` with the type `float->int->float`. This function should compute Taylor expansion of e^x around 0, of the first `n` terms. When you call your function with the arguments `taylor 0.1 3`, it should return exactly 1.105. Using `taylor 0.1 10` should produce a result close to but different from 1.105. (1) Include your Taylor function implementation below and (2) Record the result of `taylor 0.1 10` as a comment. Fill in the following:

```
let rec taylor x n =  
    (*TODO*)  
  
(*Result of taylor 0.1 10 is TODO*)
```

Tower of Hanoi (40)

First, play the game of Tower of Hanoi yourself to get an idea:

<https://www.mathsisfun.com/games/towerofhanoi.html>

After you understand the rule of the game, implement a function `move` of type

```
int -> string -> string -> string -> unit
```

so that `move n src dst aux` moves `n` disks from `src` to `dst` using `aux` as an auxillary disk.

Hint for the implementation:

- if `n` is 1, print the movement from `src` to `dst`
- otherwise, move `n-1` disks from `src` to `aux`, move 1 disk from `src` to `dst`, and move `n-1` disks from `aux` to `dst`.

- use `Printf.printf "Move from %s to %s\n"`
- for a series of expressions use `begin ... end`, e.g. `begin move...; move...; move... end.`
- You probably need to do some additional research and much try-and-error to get your ocaml code work and run.

Task: Fill in the following:

```
let rec move n src dst aux =  
  (* TODO *)
```

```
(* for testing *)  
let test () =  
  move 3 "A" "C" "B"  
  
let _ = test ()
```

How to test: Suppose the code above is in a file `hanoi.ml`, then running `ocaml hanoi.ml` will generate:

```
Move from A to C  
Move from A to B  
Move from C to B  
Move from A to C  
Move from B to A  
Move from B to C  
Move from A to C
```