Session 4: HW2 Quick Start Guidance

- First try to read `main.ml` and understand the grammar of the target language that you are going to deal with. Since we can't run `main.ml` without providing implementation of the `eval.ml`, we now try to bypass this restriction so that we can play with the grammar of the target language.
- For a quick start, add some "placeholders" to the functions you need to implement (so that the code doesn't contain any syntax error and you can run it). For example:

```
let rec fvs e = [];; (* assume there's no free variables *)
let rec subst e y m = e;; (* assume we do nothing *)
let rec reduce e = e;; (* assume we do nothing *)
```

- Then you can use the following provided sanity check cases to gradually build your implementation. See the `mytest.ml` for an example.
- Simple sanity test cases for `fvs` function:

```
(* input0 *) Var "x"
(* output0 *) ["x"]
```

```
(* input1 *) Lam("x", Var "y")
(* output1 *) ["y"]
```

```
(* input2 *) Lam("x", Lam("y", Var "z"))
(* output2 *) ["z"]
```

```
(* sample code for testing the fvs function *)

(* helper funciton for visualizing a list *)
let rec print_list = fun l ->
 match l with
  | [] -> print_string ""
  | h::t -> print_string h; print_string ", "; print_list t
;;

let my_input = Lam("x", Lam("y", Var "z"));; (* replace it with input? *)
let my_output = [];; (* replace it with output? *)
let my_eval = fvs my_input;;
print_list my_output;;
```

```
print_string "\n";;
print_list my_eval;;
print_string "\n";;
```

- Simple sanity test cases for `evaluate` function:

```
(* a simple beta-reduction case *)
(* input0 *) App( Lam("x", Var "x"), Var "y" )
(* output0 *) Var "y"
(* (\x. x) y *)
```

```
(* a different beta-reduction case *)
(* input1 *) App( Lam("x", Var "x"), Lam("y", Var "y") )
(* output1 *) Lam("y", Var "y")
```

```
(* a simple alpha-renaming case *)
(* input2 *) App( Lam("x", Lam("y", Var "x" ) ), Var "y" )
(* output2 *) Lam("v0", Var "y")
```

```
(* combining alpha-renaming and beta-reduction *)
(* input3 *) App( App( Lam("x", Lam("y", Var "x" ) ), Var "y" ), Var "z")
(* output3 *) Var "y"
```

```
(* sample code for testing the evaluate function *)
let my_input = App( Lam("x", Lam("y", Var "x" ) ), Var "y" );; (* replace it with
input? *)
let my_output = Lam("v0", Var "y");; (* replace it with output? *)
let my_eval = evaluate my_input;;
print_string (lambda_exp_2_str my_output);;
print_string "\n";;
print_string (lambda_exp_2_str my_eval);;
print_string "\n";;
```

- Notice
  - Pay attention to the conditions and operations of applying α-renaming, where the input and output should satisfy the definition of α-equivalent. You may find more details in the Wikipedia page.