CISS445: Programming Languages
Test t01

Name: _____ Score: [ ]

Open `main.tex` and enter answers (look for `answercode`, `answerbox`, `answerlong`). Turn the page for detailed instructions. To rebuild and view pdf, in bash shell execute `make`. To build a gzip-tar file, in bash shell execute `make s` and you'll get `submit.tar.gz`.

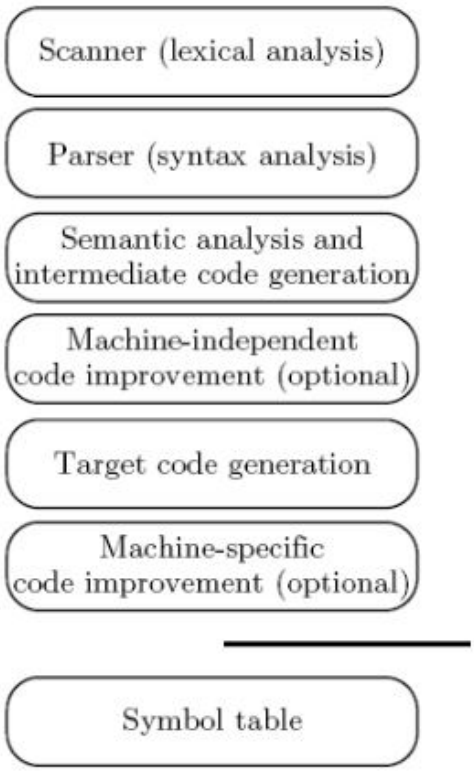
- This is a closed-book, no-calculator, no-computer, no-discussion test.
- Your solution must be written in the box provided. Anything outside the box is not considered part of the solution. Everything inside the box IS considered part of the solution.
- Do NOT provide multiple solutions. If you do, I get to pick ONE to grade. (I'm very good at picking solution – the wrong one).
- Write neatly. If I cannot read your solution easily you will get zero.
- For the OCAML coding problems, your grade will be based on passing a lists of tests cases. If your code passes all the test cases, you get all the points for that question; if it does not pass any test cases you will get 0. Not all test cases are shown below. For some questions, test cases are not shown.
- Cheating is a serious academic offense. If caught you will receive an immediate score of -100%.

Question	Points
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Question	Points
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	

TOTAL	

Q1. The following are the phases in source code compilation:



(a) What is the input for the scanner phase?
ANSWER:

(b) What is the output during the scanner phase?
ANSWER:

(c) What is the input for the parser phase?
ANSWER:

(d) What is the output during the parser phase?
ANSWER:

Q2. Write a function `quadratic` such that `(quadratic a b c)` returns a function that computes the y-value of the quadratic equation $y = ax^2 + bx + c$. For instance

```
Test                        Expected value
(quadratic 0 1 2) 5         0 * 5 * 5 + 1 * 5 + 2
(quadratic 1 2 3) 3         1 * 3 * 3 + 2 * 3 + 3
(quadratic 2 3 4) 7         2 * 7 * 7 + 3 * 7 + 4
```

ANSWER:

Q3. Write a `second` function such that `second xs` returns a list that contains exactly the second value of the list `xs`. If `xs` does not have at least two values, `[]` is returned.

Test	Expected value
`second []`	`[]`
`second [42]`	`[]`
`second [2;9]`	`[2]`
`second [3;4;5]`	`[3]`
`second [1.1; 2.2; 3.3; 4.4]`	`[2.2]`

ANSWER:

Q4. What is the value of the `x` in the following binding:

```
let x = (fun f -> (fun x -> f(f x))) (fun x -> x * x) 2;;
```

or write ERROR.

ANSWER:

Q5. Write down the type for each of the following expressions. Write ERROR if there's an error in the expression.

- Recall that the type of a function look like `'a -> 'b`. For instance the type of `fun x -> x + 1` is `int -> int` and the type of `fun x -> fun y -> x + y` is `int -> int -> int` and the type of `fun x -> fun y -> (x < y)` is `'a -> 'a -> bool`.
- The type of a list looks like `'a list`. For instance the type of `[1;2;3]` is `int list`.
- The type of a 2-tuple looks like `'a * 'b`. For instance the type of `(1, 2.4)` is `int * float`. The type for 3-tuple, 4-tuple, etc. are analgous.
- If you use type parameters, make sure you use `'a`, `'b`, `'c`, etc. in that order.

(a) `if 1 = 2 then [3] else [4]`
ANSWER:

(b) `fun x -> fun y -> y::x::[1]`
ANSWER:

(c) `fun x -> fun y -> y::[x]`
ANSWER:

(d) `fun x -> fun y -> (y (x + 1) +. 3.0)`
ANSWER:

(e) `fun x -> fun y -> y x`
ANSWER:

(f) `fun x -> fun y -> x y`
ANSWER:

Q6. Write a number guessing game. Call `(guess 42)` and the user is prompted to enter a guess. If the number entered is 42, the program will stop. If the number entered is < 42, the program will ask the user to try a larger number. If the number entered is > 42, the program will ask the user to try a smaller number.

To get an integer value from the user do this:

```
let x = read_int ();;
```

Here's a sample execute:

```
utop # guess 42;;
guess my int: 1
try higher ...
guess my int: 100
try lower ...
guess my int: 50
try lower ...
guess my int: 25
try higher ...
guess my int: 30
try higher ...
guess my int: 40
try higher ...
guess my int: 45
try lower ...
guess my int: 43
try lower ...
guess my int: 42
correct!!!
- : unit = ()
```

Turn page ...

Answer:

Q7. Write a function `reverse_at` so that `(reverse_at list n)` returns the element of the list at index `n` in the reverse order, i.e., index 0 means the last value, index 1 means the second last value, etc. Ignore the case where `n < 0` or `n >=` the size of the list.

```
Test                        Expected values
reverse_at [5;3;1] 0        1
reverse_at [2;4;6] 1        4
reverse_at [1;3;5;7] 3      1
```

ANSWER:

Q8. Write a function `rev` that reverses a list. For instance after `(rev [1; 3; 5])` is `[5; 3; 1]`. Any recursion used must be tail recursion. In other words, `(rev list)` should call another function that returns the reverse of `list` and this function computes the reverse of `list` using tail recursion.

ANSWER:

Q9. Write a function `seteq` that compares two lists are the same, treating them as sets. For instance `seteq [1;3;5] [5;3;3;1;1;3]` is true. In other words `seteq xs ys` is true exactly when the values in `xs` are in `ys` and the values in `ys` are in `xs`. You need not use tail recursion.
ANSWER:

Q10. Write a function `powerset` that computes the powerset of a list. For instance after

```
let xs = powerset [1;3;5];;
```

`xs` is `[[]; [1]; [3]; [5]; [1;3]; [1;5]; [3;5]; [1;3;5]]`. The lists need not appear in the above order. The values in a list of the above list need not be in the above order – for instance `[1;3]` can be `[3;1]`. In other words, you can view each list as a set.

ANSWER:

```

```

Q11. You are given this binary tree:

```
type 'a bintree = Empty
                | Node of ('a * 'a bintree * 'a bintree)
;;
```

(a) Write a function `has_element` such that `has_element t v` is true iff the value `v` is in the tree `t`. For instance

```
utop # let t = Node (5, Node (6, Empty, Empty), Node (1, Empty, Empty));;
val t : int bintree = Node (5, Node (6, Empty, Empty), Node (1, Empty, Empty))
utop # has_element t 5;;
- : bool = true
utop # has_element t 7;;
- : bool = false
```

ANSWER:

(b) Write a function `subtree` such that `subtree t v` is the subtree of `t` with `v` as root. For instance

```
utop # let t = Node (5, Node (6, Empty, Empty), Node (1, Empty, Empty));;
val t : int bintree = Node (5, Node (6, Empty, Empty), Node (1, Empty, Empty))
utop # subtree t 5;;
- : int bintree = Node (5, Node (6, Empty, Empty), Node (1, Empty, Empty))
utop # subtree t 6;;
- : int bintree = Node (6, Empty, Empty)
utop # subtree t 1;;
- : int bintree = Node (1, Empty, Empty)
utop # subtree t 0;;
- : int bintree = Empty
```

(You can assume that the values in the tree are unique.)
ANSWER: