CS 2041: Practice Final  
Fall 2018  
University of Minnesota  
Exam period: 20 minutes  
Points available: 40

**Background:** OCaml’s standard library has mutable, polymorphic hash table implementation which maps keys to values in the Hashtbl module which is demonstrated in a REPL nearby. Like the tree maps we created, Hashtbl provides higher-order functions for operating on the key/value associations in the map.

**Problem 1 (5 pts):** Write a function `print_all` which prints out all key/value bindings in a hash table of string/ integers. Use the higher-order function Hashtbl.iter func tbl where func is passed keys and values from the hash table and returns unit. It is demonstrated in the REPL session.

*Write your code for print_all here.*

```ocaml
1 # let table = Hashtbl.create 20;;  
2 # Hashtbl.add table "Goku" 8001;;  
3 # Hashtbl.add table "Krillin" 1770;;  
4 # Hashtbl.add table "Piccolo" 3500;;  
5 # Hashtbl.add table "Vegeta" 18000;;  
6 7 # let kpower = Hashtbl.find_opt table "Krillin";;  
8 val kpower : int option = Some 1770  
9 # let gpower = Hashtbl.find_opt table "Gohan";;  
10 val gpower : int option = None  
11 12 # use "hash_funcs.ml";;  
13 val print_all : (string, int) Hashtbl.t -> unit = <fun>  
14 val total_power : ('a, int) Hashtbl.t -> int = <fun>  
15 16 # print_all table;; (* demo print_all *)  
17 Krillin -> 1770  
18 Vegeta -> 18000  
19 Piccolo -> 3500  
20 Goku -> 8001  
21 - : unit = ()  
22 23 # total_power table;; (* demo total_power *)  
24 - : int = 31271
```

**Problem 2 (5 pts):** Write a function `total_power` which totals the values stored in a hash table with integer values. Use the higher-order function Hashtbl.fold func tbl initial where func is passed keys, values, and a running total. It is demonstrated in the REPL session.

*Write your code for total_power here.*

```ocaml
Problem 3 (5 pts): A5’s Calculon drew a distinction between a lambda expression and a closure. Describe the similarities and differences between these two things.
```
Problem 4 (10 pts): To the right is a program which makes use of lazy evaluation. Show what you expect the output for the program to be below. **Justify your answer** by describing when and how many times various outputs are printed.

```ocaml
open Printf;;
let _ =
  let exprA = lazy (printf "eval exprA\n"; 5) in
  let exprB = (printf "eval exprB\n"; 10) in
  let exprC = lazy (printf "eval exprC\n"; 15) in
  printf "AB: %d\n" ((Lazy.force exprA) + exprB);
  printf "AC: %d\n" ((Lazy.force exprA) + (Lazy.force exprC));
  printf "BC: %d\n" (exprB + (Lazy.force exprC));
;;
```

Problem 5 (5 pts): Write a function `constantly x` which creates an infinite stream which always returns the given value `x`. The function is demonstrated in the REPL session below. **Write your code for constantly here.**

```ocaml
# #use "constantly.ml";;
val constantly : 'a -> 'a Stream.t = <fun>
# let ones = constantly 1;;
val ones : int Stream.t = <abstr>
# Stream.next ones;;
- : int = 1
# Stream.next ones;;
- : int = 1
# let mines = constantly "mine";;
val mines : string Stream.t = <abstr>
# Stream.next mines;;
- : string = "mine"
# Stream.next mines;;
- : string = "mine"
```

Problem 6 (10 pts): Describe how string data may be added to A5’s Calculon language interpreter. Included in this addition would be string concatenation via the `~` operator shown below. Make sure to describe which parts of Calculon would need to be altered.

```ocaml
calculon> parsetree "hello";
Parse tree:
StrExp("hello")
calculon> def str = "hello";
str : StrDat("hello")
calculon> def hw = str ~ " world";
hw : StrDat("hello world")
```