First Exam
CS 1101 Computer Science I

Section 04, Spring 2016

Tuesday March 1, 2016
Instructor Muller

KEY

Before reading further, please arrange to have an empty seat on either side of you. Now that you are seated, please write your name **on the back** of this exam.

This is a closed-notes and closed-book exam. Computers, calculators, and books are prohibited.

- Partial credit will be given so be sure to show your work.
- Feel free to write helper functions if you need them.
- **Please write neatly.**

Problem	Points	Out Of	
1		1	
2		1	
3		2	
4		2	
5		3	(only two of 5, 6 and 7)
6		3	
7		3	
8		4	(only two of 8, 9 and 10)
9		4	
10		4	
Total		20	

1. (1 Point) For each of the following, indicate what would happen if the code was evaluated in an Python shell. If the code would produce a value, what value would it produce? If the code would produce an error, what error?

   (a)
   ```
   if (2 + 3) > 4:
     return "Boston"
   else:
     return "College"
   ```

   Answer:

   `Boston`

   (b)
   ```
   def g(z): return z * 2
   def f(x, y): return (g x, g z)

   f(2, 4)
   ```

   Answer:

   `Error: Unbound variable z.`

2. (1 Point) For each of the following, indicate what would happen if the code was evaluated in an Python shell. If the code would produce a value, what value would it produce? If the code would produce an error, what error?

   (a)
   ```
   def f(x, y):
     def g(x): return x + y
     return g(x + y)

   f(3, 4)
   ```

   Answer:

   `11`

   (b)
   ```
   def g(x, y, z): return x + y
   def f(x, y): return g()

   f(1, 6)
   ```

   Answer:

   `Error: wrong number of arguments to g.`

3. (2 Points) Write a function `bump : int -> int` that doubles odd numbers and triples even numbers. For example, the call `bump(14)` should evaluate to `42` while the call `bump(15)` should evaluate to `30`.

Answer:

```
# bump : int -> int
#
def bump(n):
  if (n % 2) == 0:
    return n * 3
  else:
    return n * 2
```

4. (2 Points) A customer is eligible for a discount if they are under 21 years of age and they have a gold card or if they are between the ages of 60 and 90. (Ninety one? You're out of luck.) Write a function `eligible : bool * int -> bool` such that a call `eligible(goldCard, age)` returns `True` if they are eligible for a discount and `False` otherwise.

Answer:

```
# eligible : bool * int -> bool
#
def eligible(goldCard, age):
  return (goldCard and age < 21) or (age >= 60 and age <= 90)
```

3 Point Problems – choose only two

Circle the numbers of the two problems that you want graded.

5. (3 Points) The built-in `%` operator computes the integer remainder. In particular, the expression `m % n` evaluates to the integer remainder when `m` is divided by `n`. Write `mod` as a function `mod : int * int -> int` such that a call `mod(m, n)` evaluates to the integer remainder when `m` is divided by `n`. Your solution should not use the built-in operator of the same name. For the purposes of this problem, you may assume that `m` and `n` are non-negative. Hint: consider repeated subtraction.

   Answer:

```
# mod : int * int -> int
#
def mod(m, n):
  if m < n:
    return m
  else:
    return mod(m - n, n)
```

6. (3 Points) Write a function `snds : (A * B) list -> B list` such that a function call `snds(pairs)` returns a list of the second components of the pairs. For example, the call `snds([(1, 'A'), (2, 'B')])` should return the list `['A', 'B']`.

   Answer:

```
def snds(pairs):
  return map((lambda p : p[1]), pairs)
```

7. (3 Points) Write a function `rotateLeft : int * 'a list -> 'a list` such that a call `rotateLeft(n, xs)` rotates `xs` leftward `n` positions. By "rotate" we mean that elements that fall off the left end, migrate to the right end. For example, the call `rotateLeft(3, ['A', 'B', 'C', 'D'])` should evaluate to the list `['D', 'A', 'B', 'C']`. You may assume that `n` is non-negative.

Answer:

```
# rotateLeft : int * 'a list -> 'a list
#
def rotateLeft(n, xs):
  if n == 0:
    return xs
  else:
    first = xs[0]
    rest = xs[1:]
    return rotateLeft(n - 1, rest + [first])
```

4 Point Problems – choose only two

Circle the numbers of the two problems that you want graded.

8. (4 Points) It's Super Tuesday and time to cull the field for the next debate. The debate organizers have analyzed the polls and gathered summary data in a list of pairs:

```
candidates = [("Trump", 0.30), ("Rubio", 0.20), ("Cruz", 0.18), ("Carson", 0.02)]
```

The debate organizers wish to invite only candidates with above average poll numbers. Write a function

```
cull : (string * float) list -> (string * float) list
```

such that a call `cull(candidates)` returns the list of those candidates with above average poll numbers. For example, on the data above, since the average is **0.175**, the **cull** function would return the list of three candidates `[("Trump", 0.30), ("Rubio", 0.20), ("Cruz", 0.18)]`.

Answer:

```
# cull : (string * float) list -> (string * float) list
#
def cull(candidates):
  ave = average(candidates)
  return filter(lambda p : p[1] > average, candidates)

def average(candidates):
  numbers = map(lambda p : p[1], candidates)
  return reduce(operator.add, numbers, 0) / len(numbers)
```

9. (4 Points) We take our positional numeral system for granted but it was at least a couple of millennia in the making. The "positions" represent powers of 10 ascending from right to left: ones, tens, hundreds and so forth. For example, the decimal numeral for the present year, 2016, can be understood as

$$\begin{aligned} 2016_{10} &= 2 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 6 \times 10^0 \\ &= 2 \times 1000 + 0 \times 100 + 1 \times 10 + 6 \times 1 \\ &= 2000 + 0 + 10 + 6 \\ &= 2016 \end{aligned}$$

The system is so robust and flexible that it works for any base. For example, a numeral in base 3 can be understood in the same way:

$$\begin{aligned} 201_3 &= 2 \times 3^2 + 0 \times 3^1 + 1 \times 3^0 \\ &= 2 \times 9 + 0 \times 3 + 1 \times 1 \\ &= 18 + 0 + 1 \\ &= 19_{10} \end{aligned}$$

Write a function `toDecimal : int list * int -> int` such that a call `toDecimal(digits, base)` returns the decimal value of the list of digits. For example, the call `toDecimal([2, 0, 1], 3)` should return `19`. Feel free to use the `a.reverse()` function which reverses a list **a**.

Answer:

```
# toDecimal : int list * int -> int
#
def toDecimal(digits, base):
  def repeat(digits, power, acc):
    if digits == []:
      return acc
    else:
      digit = digits[0]
      digits = digits[1:]
      n = digit * (base ** power)
      return repeat(digits, power + 1, n + acc)

  return repeat(List.rev digits, 0, 0)
```

10. (4 Points) Given a decimal numeral X and base b, how can the numeral X_{10} be converted to the equivalent numeral in base b? How might we write a function `decimalTo : int * int -> int list` so that a call such as `decimalTo(19, 3)` would evaluate to the list of digits `[2, 0, 1]` making up the base 3 numeral 201_3?

Let's say we have a helper function:

```
def div(m, n): return (m / n, m \% n)
```

which computes both the integer quotient and the remainder when m is divided by n. Then we can proceed in a repetitive process as follows:

$$\begin{aligned}\texttt{div}(19,3) &= (6,1)\\ \texttt{div}(6,3) &= (2,0)\\ \texttt{div}(2,3) &= (0,2)\end{aligned}$$

In each iteration from one line to the next, the quotient of the previous step moves down to the left and the successive remainders make up the digits of the answer from right to left. When the quotient is zero the process is complete. Write the function `decimalTo`.

Answer:

```
# decimalTo : int * int -> int list
#
def decimalTo(n, base):
  def repeat(n, acc):
    if n == 0:
      return acc
    else:
      (q, r) = div(n, base)
      return repeat(q, [r] + acc)

  return repeat(n, [])
```