First Exam
CS 101 Computer Science I

KEY
Thursday October 9, 2014
Instructor Muller
Boston College
Fall 2014

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-book exam. Computers, calculators, and books are prohibited. You may use one 8.5 by 11 sheet of notes. **Please choose between problems 7 and 8, circling the number of the one you want graded.** In solving problems involving repetition, you are free to use any form that you would like. Partial credit will be given so be sure to show your work. **Please try to write neatly.**

Problem	Points	Out Of
1		3
2		2
3		2
4		2
5		3
6		3
7		5 (one of)
8		5 (one of)
Total		20

1. (3 Points) For each of the following, indicate what would happen if the code was evaluated in a 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): return (g(x), g(y))
   def g(z): return z * 2
   f(2, 4)
   ```
   Answer:
   ```
   (4, 8)
   ```

   (b)
   ```
   def f(x, y):
     def g(x): return x + y
     return g(x + 1)
   f(1, 6)
   ```
   Answer:
   ```
   8
   ```

   (c)
   ```
   def f(x, y): return g()
   def g(): return x + y
   f(1, 6)
   ```
   Answer:
   ```
   Error: unbound variable y
   ```

2. (2 points) The word *millennial* is sometimes used to refer to a person born after January 1, 1981 and before December 31, 1996. Write a function `isMillennial : int -> bool` that accepts a birth year and returns `True` if the person is a millennial. Otherwise it should return `False`.

   Answer:
   ```
   # isMillennial : int -> bool
   #
   def isMillennial(birthYear):
     return (1981 <= birthYear) and (birthYear <= 1996)
   ```

3. (2 points) *Compounded growth* of a quantity is usually expressed as a periodic percentage together with the number of periods of growth. In particular, a present value pv growing at i percent per period will grow to pv$(1+i)^n$ over n periods. Write a function `futureValue : float * float * float -> float` such that a call `futureValue(pv, i, n)` will compute the compounded growth over `n` periods.

Answer:

```
# futureValue : float * float * float -> float
#
def futureValue(pv, i, n):
  return pv * (1.0 + i) ** n
```

4. (2 Points) Write a Python function `downFrom : int -> int list` such that a call `downFrom(n)` produces the list `[n - 1, n - 2, ..., 0]`. For example, the call `downFrom(5)` should evaluate to the list `[4, 3, 2, 1, 0]`.

Answer:

```
 downFrom : int -> int list                                PRODUCE A LIST
#
def downFrom(n):
  return [ n - i - 1 for i in range(n) ]
```

5. (3 Points) Write a Python function `interesting : a * (a * a -> bool) * a list -> int` such that a call `interesting(x, test, xs)` returns the number of elements of `xs` that are in the `test` relation with `x`. For example, assuming that the built-in `operator.lt` (less than) function is imported, the expression `interesting(5, operator.lt, [3, 4, 5, 6, 7])` would evaluate to 2 because only list elements 3 and 4 are less than 5.

   Answer:

```
# interesting : a * (a * a -> bool) * a list -> int
#
# The call interesting(x, test, xs) returns the number of elements
# of xs that are in the test(., x) relation with x.
#
def interesting(x, test, xs):
  if xs == []:
    return 0
  else:
      n = interesting(x, test, xs[1:])
      if test(xs[0], x):
        return n + 1
      else:
        return n
```

6. (3 Points) Write a Python function `removeNth : int * a list -> a list` such that a call of the function `removeNth(n, xs)` returns a list that is just like `xs` but the `n`th element has been removed. For example, the call `removeNth(0, [10, 20, 30])` should evaluate to `[20, 30]` while the call `removeNth(2, [10, 20, 30])` should evaluate to `[10, 20]`. You may assume that the list has an `n`th element.

   Answer:

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

7. (5 Points) Write a Python function `middle : int list -> int` that accepts an odd-length list of unique integers and returns the middle integer. For example, the call `middle([20, 18, 50, 62, 30])` should evaluate to `30` because there are 2 values less than `30` and 2 values greater. The call `middle([10])` should evaluate to `10`.

Answer:

```
# middle : int list -> int                                CONSUME A LIST
#
# The call middle(ns) accepts an odd-length list of unique integers. It
# returns the one that has an equal number lower and higher.
#
def middle(lst):
  triples = [ (i, interesting(i, lt, lst), interesting(i, gt, lst)) for i in lst ]

  def loop(triples):
    if triples == []:
      return None
    else:
      (i, smaller, bigger) = triples[0]
      if smaller == bigger:
        return i
      else:
        return loop(triples[1:])
  return loop(triples)
```

8. (5 Points) If **pic** is a `stddraw.Picture`, the function call `pic.filledRectangle(x, y, hW, hH, color)` will add a filled rectangle to `pic` centered at `(x, y)` with width twice `hW`, height twice `hH` and of color `color`.

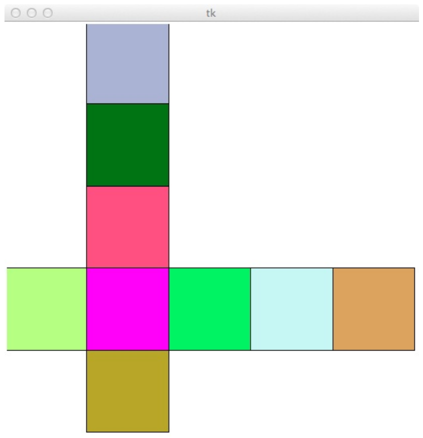
Write a function `cross : Picture * int * int -> void` such that a call `cross(picture, n, i)` will render randomly colored squares of side `1.0 / n` across row `i` and up column `i` of the unit square. For example, executing the code

```
import stddraw

def testCross():
  myPic = stddraw.Picture()
  cross(myPic, 5, 2)
  myPic.start()

testCross()
```

would produce the following picture in the graphics window:



Answer:

```
def cross(picture, n, i):
  side = 1.0 / n
  halfSide = side / 2.0
  fixedXY = (i - 1) * side + halfSide
  def loop(m):
    if m > 0:
      varyingXY = (m - 1) * side + halfSide
      color = picture.randomColor()
      picture.filledRectangle(fixedXY, varyingXY, halfSide, halfSide, color)
      color = picture.randomColor()
      picture.filledRectangle(varyingXY, fixedXY, halfSide, halfSide, color)
      loop(m - 1)
  loop(n)
```