First Exam
CS 101 Computer Science I

KEY
Tuesday October 1, 2013
Instructor Muller
Boston College
Fall 2013

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

This is a closed-book and closed-notes exam. Computers, calculators, books and notes are prohibited. 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		4
2		3
3		5
4		5
5		4
6		5
7		3 (extra credit)
Total		26

1. (4 Points) Consider the following (somewhat oddly spaced) Python function.

```
def f( n ):
      ---
  def loop( m , answer ):
           --- --------
    if m == 0:
      return answer
    else:
      return loop( m - 1 , m * answer )

  return loop( n , 1 )
```

   (a) (2 points) Underline all occurrences of **formal parameters** and circle all occurrences of **actual arguments**.

   (b) (2 points) Use equations and the **replacement model** to show the value of the expression **f(1 + 2)**. Feel free to omit the keyword "return" in your answer.

   Answer:

```
f(1 + 2) = f(3)
         = loop(3, 1)
         = loop(3 - 1, 3 * 1)
         = loop(2, 3)
         = loop(2 - 1, 2 * 3)
         = loop(1, 6)
         = loop(1 - 1, 1 * 6)
         = loop(0, 6)
         = 6
```

2. (3 Points) Python's built-in **int** function returns the integer part of a floating point number. For example, the value of **int(2.8)** is **2**. Write a function **fractionPart : float → float** such that a call **fractionPart(number)** returns just the fractional part. For example, the call **fractionPart(2.8)** should return **.8**.

   Answer:

   def fractionPart(number): return number - int(number)

3. (5 Points) If **pic** is a **stddraw.Picture**, the function **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**. In Part A of problem set 2, you wrote a function **ring(picture, x, y, radius, width, color)** which added a colored ring to **picture**. Write a function

   frame : Picture * float * float * float * float → void

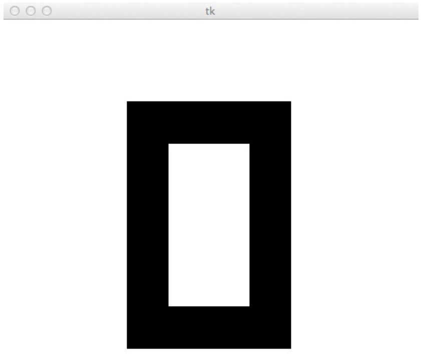
   such that a call **frame(picture, x, y, halfWidth, halfHeight, trimSize)** will add a black frame to **picture** centered at **(x, y)**, with total width twice **halfWidth**, total height twice **halfHeight** and with a uniform width black trim of size **trimSize**. For example, executing the code

```
import stddraw

def testFrame():
  myPic = stddraw.Picture()
  frame(myPic, .5, .5, .2, .3, .1)
  myPic.start()

testFrame()
```

   would produce the following picture in the graphics window:



   Answer:

```
def frame(picture, x, y, hW, hH, trim):
    picture.filledRectangle(x, y, hW, hH, 'black')
    picture.filledRectangle(x, y, hW - trim, hH - trim, 'white')
```

4. (5 Points) In Python, the ** operator performs exponentiation. For example, **2 ** 3** evaluates to **8**. Assume that that ** operator doesn't exist and write a function **pow : int * int → int** such that a call **pow(m, n)** returns the value m^n. Of course, anything raised to the 0 power is **1**.

   One point extra credit for writing your **pow** function **tail-recursively**.

   Answer:

```
def pow(m, n):
  if n == 0:
    return 1
  else:
    return m * pow(m, n - 1)

def pow(m, n):                         Tail-recursive version
  def repeat(n, answer):
    if n == 0:
      return answer
    else:
      return repeat(n - 1, m * answer)
  return repeat(n, 1)
```

5. (4 Points) Write a Python function **goodBMI : int * float → bool** such that a call **goodBMI(age, bmi)** returns **True** if **bmi** is a good Body-Mass Index for someone of age **age**. Your function should return **False** if **bmi** isn't a good BMI for someone of **age**. For the purposes of this problem, let's say that a BMI between 24.0 and 20.0 (inclusive) is good for someone under age 25. For someone over age 25, a good BMI must be between 27.0 and 21.5.

Answer:

```
def goodBMI(age, bmi):
  if age < 25:
    return (bmi >= 20.0) and (bmi <= 24.0)
  else:
    return (bmi >= 21.5) and (bmi <= 27.0)
```

6. (5 Points) Write a Python function **allFactors : int * (int list) → bool** such that a call **allFactors(m, listOfInts)** will return **True** if and only if every number in **listOfInts** is an integer factor of **m**. For example, the calls **allFactors(20, [4, 5])**, **allFactors(16, [2, 4, 8])** and **allFactors(17, [])** should all return **True** while **allFactors(20, [4, 5, 6])** should return **False**.

Answer:

```
def allFactors(n, listOfInts):
  if listOfInts == []:
    return True
  else:
    return isFactor(listOfInts[0], n) and allFactors(n, listOfInts[1:])
```

7. (3 Point Extra Credit Challenge Problem) Many computer applications (such as Excel) use floating point numbers to represent dates and times. For times, for example, the idea is to use the digits to the left of the decimal place to represent the hour and the digits to the right of the decimal place to represent the number of minutes as a percentage of 60. Using this scheme, the number **6.5** would represent **6:30AM**.

   Write a Python function **time : float** → **string** such that a call **time(num)** returns a string representation of the time using 12-hour time. For example, the call **time(6.0)** should return the string **'6:00AM'**, the call **time(6.5)** should return the string **'6:30AM'**, the call **time(13.9)** should return the string **'1:54PM'** and the call **time(25.0)** should return the string **'1:00AM'**. You may assume that the argument provided to **time** is non-negative.

   Answer:

```
def time(number):
  preMinutes = int((number % 1.0) * 60)
  minutes = ("0" if preMinutes < 10 else "") + str(preMinutes)

  oneDay = int(number) % 24
  meridian = "AM" if oneDay < 12 else "PM"
  preHour = oneDay % 12
  hour = str(preHour) if preHour != 0 else "12"
  return hour + ":" + minutes + meridian
```