Second Midterm Exam
CSCI 1100 Gateway to Computer Science

KEY
Thursday November 17, 2022
Instructor Muller
Boston College
Fall 2022

Please do not write your name on the top of this test. Before reading further, please arrange to have an empty seat on either side of you. Now that you are seated, please note the number on top of your test and write it together with your name on the sheet that is circulating.

This is a closed-book and closed notes exam. For problems involving repetition, feel free to use a solution to one problem in solving another problem. And feel free to use any repetition 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		6
3		10
Total		20

1 Snippets (4 Points Total)

For each of the following code snippets, indicate what would happen when attempting to load and then run them. If they have a problem that would be detected when loaded, indicate the problem. If they have a problem that would be detected when run, indicate the problem. If they have no problems, indicate what value they would produce.

1. (1 Point)

```
def h(a, b):
  a * b = c
  return c

h(4, 5)
```

Answer: Error cannot assign to a * b

2. (1 Point)

```
def f(n):
  return "Even" if n % 2 == 0 else "Odd"

f(4)
```

Answer: Program is fine, returns "Even"

3. (1 Point)

```
def mystery(x, y):
  return x + y

mystery(2, 3, 4)
```

Answer: Too many arguments in the call

4. (1 Point)

```
def f(m, n):
  return [[ i + j for i in range(m) ] for j in range(n) ]

f(2, 3)
```

Answer: Code is OK, returns `[[0, 1], [1, 2], [2, 3]]`.

2 Fluency with Repetition Idioms (6 Points)

1. (3 Points) Rewrite the following `rower` function without using a list comprehension.

```
def rower(rows, cols):
  return [[ row for _ in range(cols)] for row in range(rows)]
```

Answer:

```
def rower(rows, cols):
  ans = []
  for row in range(rows):
    oneRow = []
    for _ in range(cols):
      oneRow.append(row)
    ans.append(oneRow)
  return ans
```

2. (3 Points) The `many` function copies a string. In particular, a call `many(n, string)` returns a string with `n` copies of `string`. For example, the call `many(2, "Go BC! ")` returns the string `"Go BC! Go BC! "`. Rewrite the `many` function without using a `for`-loop.

```
def many(n, string):
  ans = ""
  for _ in range(n):
    ans = ans + string
  return ans
```

Answer:

```
def many(n, string):
  ans = ""
  while n > 0:
    ans = ans + string
    n = n - 1
  return ans
```

3 Repetition (10 Points Total)

1. (5 Points) Python has a built-in type `set` but for this problem we'll use simple lists as sets of integers. As you know, a set has no duplicates. So `[1, 3, 2]` is a good set but `[1, 3, 1]` isn't. Write a function `intersection : list int * list int -> list int` such that a call `intersection(s1, s2)` returns the set if integers that are in both `s1` and `s2`. For example, the call `intersection([1, 3, 2], [2, 3, 4])` should return the list/set `[3, 2]` because 2 and 3 are in both sets. Note that the `intersection` function returns `[]` if either or both of `s1` and `s2` are empty.

Answer:

```
def intersection(s1, s2):
  ans = []
  for i in s1:
    if i in s2:
      ans.append(i)
  return ans
```

2. (5 Points) Humans use base 10 but digital computers traffic in base 2 and base 16. This question relates to conversion between base 10 and any other base. First consider the simple function `div`

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

Given `m` and `n`, the `div` function returns both the integer quotient and the integer remainder when `m` is divided by `n`. If we want to convert 25_{10}, to base 3, say, we note

```
div(25, 3) = (8, 1)
div(8,  3) = (2, 2)
div(2,  3) = (0, 2)
```

And low and behold $25_{10} = 221_3$. How so? Well

$$221_3 = (2 * 3^2) + (2 * 3^1) + (1 * 3^0) = (2 * 9) + (2 * 3) + (1 * 1) = 18 + 6 + 1$$

Write a function `decimalTo : int * int -> list int` such that a call `decimalTo(number, base)` will return the list of digits forming the `number` in the new `base`. In the example above, the call `decimalTo(25, 3)` would return the list `[2, 2, 1]`.

Answer:

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

def decimalTo(number, base):
  answer = []
  while number > 0:
    (number, remainder) = div(number, base)
    answer.append(remainder)
  return answer.reverse()
```