Test1 Version A

Important notes:

1. Do not start until you are instructed to do so!
2. We suggest you do not use your browser's zoom feature. Instead...
    - Click near left edge to make font smaller.
    - Click near right edge to make font bigger.
3. You will have 55 minutes once the proctor says to start.
4. You will have a few minutes of additional time after we stop to scan and submit your solutions.

5. Just before the quiz...
    1. Have a fully-charged smartphone and laptop, and still plug both in if possible
    2. Log into Gradescope on your phone
    3. Change the screen timeout setting on your phone to never, so your screen doesn't go black if you don't interact with your screen for a while.
        - iPhones: Settings / Display & Brightness / Auto-Lock / Never
        - Android: Settings / Display / Screen timeout / 10 minutes (or the maximum amount of time)
    4. Turn on Do Not Disturb (or otherwise turn off all notifications).
    5. Position your webcam so we can see:
        - Your desk

- The paper you are working on
- Your writing utensil(s)
- Both of your hands
- Your phone

6. During the quiz:
    1. You may not ask questions during the exam.
        - If you are unsure how to interpret a problem, take your best guess.
    2. You may not touch your laptop or webcam.
        - This includes muting yourself at any point; the proctors may mute you though.
    3. All of these must be visible at all times:
        - Your desk
        - The paper you are working on
        - Your writing utensil(s)
        - Both of your hands
        - Your phone, with the quiz webpage
    4. For any tech fails (laptop or internet stops working, etc.):
        1. Stop taking the quiz
        2. Immediately email pokade@andrew.cmu.edu
        3. We will email you soon to set up a 1-on-1 oral quiz with the course faculty

7. After the quiz:
    1. Follow all proctor instructions on how to end the quiz.
    2. Keep everything in view (as noted above) until the proctor calls "time".
    3. When instructed, use your phone to scan your quiz and submit the PDF to Gradescope.
    4. After submitting to Gradescope, hold your phone up to the webcam to show the receipt.

5. Even then, remain in quiz mode until the proctor calls "all clear"

---

1. Short Answer [10 points]

1. List two course resources or events where you can get help on an assignment.
2. True or False: The term project has less grade weight than the final.
3. Austin wrote a program with a syntax error, runtime error, and logical error in it. Which error did Python raise first?
4. If s is a non-empty string and `s[s.find('z')] != 'z'` what must be true in general about s?
5. What is `-7 // 3`? What is `-7 % 3`?
6. Solve this one-liner RC: `def rc(n): return round(n) != roundHalfUp(n)`

---

2. areAmicableNumbers(x, y) [11 points]

Definitions:

1. The proper divisors of n are the factors of n, including 1, not including n.
2. Numbers x and y are amicable numbers if the sum of the proper divisors of x equals y and the sum of the proper divisors of y equals x.

Examples:

1. 220 and 284 are amicable numbers because the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110, which sum to 284. The proper divisors of 284 are 1, 2, 4, 71 and 142, which sum to 220.

With this in mind write the function areAmicableNumbers(x, y) which takes in two non negative integers x and y and returns True if they are amicable numbers and False if they are not.

---

3. nthMarriedNumber(n) [21 points]

Definitions:

1. A number n is a married number if it can be split into two amicable numbers

Examples:

1. 220284 is a married number because it can be split into 220 and 284.
2. 6006 is a married number because it can be split into 6 and 006

With this in mind write the function nthMarriedNumber which takes a non negative integer n and returns the nth married number.

The first few married numbers are: 66, 606, 2828, and 6006

You may assume that areAmicableNumbers has already been written and use it as a helper, even without completing areAmicableNumbers.

We encourage you to use good top down design. While not required, it may be helpful to write:

1. digitCount(n)
2. splitNumber(n, i) such that splitNumber(123, 1) == (12, 3) and splitNumber(12345, 3) == (12, 345)
3. isMarriedNumber(n)

You may not use strings, lists, or recursion on this problem. In particular, any solution that uses strings will receive no credit.

---

4. runStringProgram(s) [21 points]

Definitions:

1. s is a string of the form "data:program"
2. data is a string of letters
3. program is a period-separated list of commands of the form 'Sxyz', 'Sxy', or 'L?'
   - The command 'Sx' sets data = data[:x]
   - The command 'Sxy' sets data = data[x:y]
   - The command 'L?' causes runStringProgram() to return True if data is all lowercase letters, or to just move on to the next command if not
4. If we finish running the program without returning True from an 'L?' command we return False

Examples:

1. `runStringProgram("aaaBBBccc:S6.L?.S13.L?")`
   - data = "aaaBBBccc"

- after command 'S6', data = data[:6] = "aaaBBB"
- L? does not cause runStringProgram() to return True since there are capital Bs
- after command S13, data = data[1:3] = "aa"
- L? causes runStringProgram() to return True

2. `runStringProgram("qPxYqP:S24.L?")`
   - data = "qPxYqP"
   - after command S24, data = data[2:4] = "xY"
   - L? does not cause runStringProgram() to return True since there is a capital Y
   - Since we finished the program without returning True, we return False

With this in mind, write the function runStringProgram(s) which splits s into our data and commands, runs the commands, returns True if our string program should return True, and returns False otherwise.

You may not use lists or recursion on this problem. In particular, you may not index into or assign variables directly to the result of .split() (so `a = s.split(c)[0]` is disallowed), but you may loop over its results.

---

5. Code Tracing [11 points]

What does the following code print?
Be certain to show your work, and also very clearly circle your answer!

```
def f(s):
    a = s[0]
    b = s[-1]
```

```
    return chr(ord(a) + int(b))

def ct1(s):
    t = ''
    for i in range(len(s)):
        if s[i].isupper():
            t += f(s[i:i+2])
        elif s[i].isdigit():
            t += s[i-1:i+2:2][::-1]
        else:
            t += '.\n'
    return t

print(ct1("aB1C2xD3"))
```

---

6. Code Tracing [15 points]

What does the following code print?
Be certain to show your work, and also very clearly circle your answer!

```
def ct2(x):
    for y in range(x, x+2):
        for z in range(x, x-2, 2):
            print("A", z, end=" ")
        for z in range(5,6):
            x += 1
            if ((x + z) % 2 == 0):
                print("B", x, end=" ")
        for z in range(y, -1, -2):
            if z <= y-3:
```

```
                continue
            print("C", z, end=" ")
ct2(3)
```

---

7. Reasoning Over Code [11 points]

What input makes the following function return True? Be certain to show your work, and also very clearly circle your answer!

```
def f(n):
    a = 0
    while n > 0:
        n //= 10
        a += 1
    return a

def g(n):
    return int(n**0.5)**2 == n

def rc(n):
    assert(n > 99999)
    while n > 0:
        a = n % 10
        b = n // 10**(f(n)-1)
        if not g(10*b+a):
            return False
        n //= 10
        n %= 10**(f(n)-1)
    return True
```

---

8. Bonus [2 points]

What input makes the following function return True? Be certain to show your work, and also very clearly circle your answer!

```
def f(n, c):
    if n < 0: return c
    return f(n-1, c+1/2**n)

def bonusRC(n):
    return almostEqual(n, f(100,0))
```
