Exam guidelines

Please read these guidelines carefully.

1. You must write every single piece of code in your program using your own understanding. Any suspicion of copying and pasting (from any source) will be treated as a violation of academic integrity.

2. You can work on the initial scaffolds with one or two close classmates. Once you reach the code process, you must complete the work alone.

3. No large-scale collaboration (more than three people) allowed.

4. You may use Google as a reference in addition to our normal class materials such as PY4E and Dr. Chuck's videos.

Please review the UCR Integrity Statement.

At the University of California, Riverside honesty and integrity are fundamental values that guide and inform us as individuals and as a community. The academic culture requires that each student take responsibility for learning and for producing work that reflect their intellectual potential, curiosity, and capability. Students must represent themselves truthfully, claim only work that is their own, acknowledge their use of others' words, research results, and ideas, using the methods accepted by the appropriate academic disciplines and engage honestly in all academic assignments. Misunderstanding of the appropriate academic conduct will not be accepted as an excuse for academic misconduct. If a student is in doubt about appropriate academic conduct in a particular situation, he or she should consult with the instructor in the course to avoid the serious charge of academic misconduct.

To ensure the highest standard of academic integrity, all students should be familiar with the guidelines posted at: http://conduct.ucr.edu/docs/generalconductbrochure.pdf

BUS 175 20S
Exam 1: Alice and Bob

Goal

Use any material covered so far (up to and including Week 5: Functions) to write a cool program!

Background

Alice wants to send and receive messages with Bob. But they have a problem: a third person, Charlie, often reads their messages and shares them with the world. Alice needs a solution to turn her messages into scrambled text that she can send to Bob. Bob can also use this solution to descramble the text back into a message.

You heard on a history podcast once that people used to scramble messages by shifting each letter by a number N (the key) known by the sending and receiving party. For example, the letter "a" shifted forward by three becomes the letter "d" — a → b → c → d. To descramble "d", shift it backward by three — d → c → b → a. You then think of something clever: What if the second character in the message was shifted by (N + 1), and the third character was shifted by (N + 2), and so on? This would help ensure extra security of the message so Charlie can't understand it. You also decide to shift numbers 0 – 9 in addition to the letters a – z.

You propose this solution to Alice and she really likes your thinking. She asks that you present the final program to her soon so she can continue to message with Bob.

What your Python program should do

Ask the user to enter the action (scramble or unscramble), the message, and the key N. The entry must follow a very specific format. First, the message must start with "enc" (for encrypt or scramble) or "dec" (for decrypt or descramble). This is followed by the characters "<>" (the normal less than and greater than keys). Next comes the text to be encrypted / decrypted. This is followed by another set of "<>". Finally, at the end is the key N. All of this should be entered in one line / entry.

Spaces and symbols are allowed in the message and count as a character. Anything that is not a letter (a – z) or number (0 – 9) can stay the same and does not need to be shifted.

Once you receive the user's input, first print the text that was entered, then print the new text. The user should be able to enter messages as many times as desired until they enter "q" to quit.

Please pay very close attention to the walkthrough output below to learn how this program should work.

Walkthrough output

```
Please enter your message, or q to quit:
enc<>a b c<>5
```

Note the syntax of the entry. Spaces and symbols are allowed in the message.

```
Your original message is: a b c
```

Your first output should be just the text of the message.

2. "b" shifted forward by seven is "i". Remember that N increases by one for each character in the message. This means that for the space between "a" and "b", N is six (but spaces don't get shifted). Once you get to "b", N is seven.

```
Your secret message is: f i l
```

1. "a" shifted forward by five is "f".

3. "c" shifted forward by nine is "l".

```
Please enter your message, or q to quit:
enc<>dogs<>10
Your original message is: dogs
Your secret message is: nzs5
```

"s" shifted forward by 13 (why?):
s → t → u → v → w → x → y → z → 0 → 1 → 2 → 3 → 4 → 5
Don't forget to start the numbers at 0!

```
Please enter your message, or q to quit:
dec<>nzs5<>10
```

Now we are decrypting the message, or turning scrambled text into the readable message.

```
Your secret message is: nzs5
```

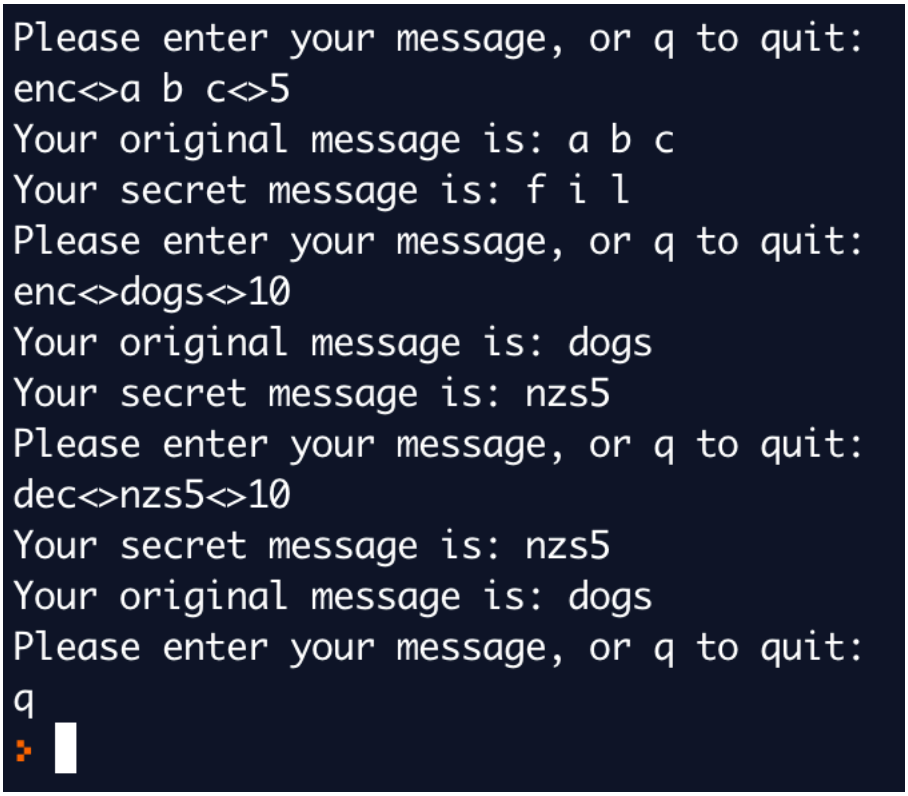
Shift "n" backward by ten, "z" backward by 11, "s" backward by 12, and "5" backward by 13. N increases by one for each character, no matter which way you are shifting.

```
Your original message is: dogs
Please enter your message, or q to quit:
q
```

Sample output

The walkthrough output above was broken up to allow room for the explanatory text boxes. Here is what the real output should look like in both plain text and repl form.

```
Please enter your message, or q to quit:
enc<>a b c<>5
Your original message is: a b c
Your secret message is: f i l
Please enter your message, or q to quit:
enc<>dogs<>10
Your original message is: dogs
Your secret message is: nzs5
Please enter your message, or q to quit:
dec<>nzs5<>10
Your secret message is: nzs5
Your original message is: dogs
Please enter your message, or q to quit:
q
```



Assumptions and clarifications

- Assume all input will be lowercase.
- The order of the letters and numbers is a, b, c, d, … x, y, z, 0, 1, 2, … 8, 9, and then back around to a, b, c, and so on.
- If the user input does not begin with "enc" or "dec", ask the user for input again.
- When the action is "enc", shift forward (i.e., a to z order).
- When the action is "dec", shift backward (i.e., z to a order).
- Spaces and symbols are allowed in the message. They are treated as characters (for the purposes of increasing N) but do not get shifted. They will remain the same in the new message. Please see the additional sample output.

If you have any questions about what can or should happen, please check and post on Campuswire!

Additional sample output

The walkthrough output above was broken up to allow room for the explanatory text boxes. Here is what the real output should look like in both plain text and repl form.

```
Please enter your message, or q to quit:
this is not the right way to enter a message
Please enter your message, or q to quit:
enc<>tacos-are.tasty<>5
Your original message is: tacos-are.tasty
Your secret message is: ygjw1-l3r.8q9bh
Please enter your message, or q to quit:
enc<>spaces and symbols don't get shifted<>17
Your original message is: spaces and symbols don't get shifted
Your secret message is: 97twze yc3 krg6kiq dpp'x ml1 2sus7tt
Please enter your message, or q to quit:
enc<>12345<>10
Your original message is: 12345
Your secret message is: bdfhj
Please enter your message, or q to quit:
dec<>bdfhj<>10
Your secret message is: bdfhj
Your original message is: 12345
Please enter your message, or q to quit:
enc<>prof. rich<>8
Your original message is: prof. rich
Your secret message is: x0yq. 5xsy
Please enter your message, or q to quit:
q
```

How to work on and submit your Python code

Please follow the same instructions with repl as you have done throughout the class.

Deliverables

Submit your repl share link to iLearn *before the due date and time* listed on the iLearn submission link.

Grading Rubric

200 points total.