SY201 – Cyber Operations Fundamentals; Programming Assignment `04`

Password Validator

Name: _____ Alpha: _____

Begin this assignment by making sure your repo is up to date. Change to **`~/repo201`** and type:

**`git pull`**

All the files required for this assignment, including an electronic version of this handout, are available in:

**`~/repo201/programming/pa04`**

---

*Read this entire handout at least once before you begin your research or start coding*

1. **Background**:

In this assignment you will create a program that reads passwords from a file and determines if each one meets a minimum set of requirements for length and complexity.

2. **Your Program Should Do This:**

   a. Prompt the user for the name of a file containing passwords.

   b. Prompt the user for the name of a file containing the minimum specifications for the passwords in the first file.

   c. Passwords can only contain letters, numbers, digits and special characters with ASCII values `33` through `47`; `58` through `64`; and `91` through `96`.

   d. There will be exactly five lines in the specification file. Each line represents a different minimum requirement for passwords. From top to bottom, the lines are:

      i. Minimum number of total characters
      ii. Minimum number of uppercase letters [`A-Z`]
      iii. Minimum number of lowercase letters [`a-z`]
      iv. Minimum number of digits [`0-9`]
      v. Minimum number of special characters

e. Read each line from the password file (each line is a single password) and determine if the password meets the minimum complexity requirements that you established from the specification file.

f. For each password you process, print its number, the password itself, and whether it is *Valid* or *Invalid*. When you've processed all the passwords in the file, and printed their status, print a final summary of your findings.

3. **Design Requirements:**

a. Think carefully about how you want to approach this assignment before you start coding. It's rarely effective when you open up your editor and just start typing. Think first about the kinds of variables you will use, their names, and how the different sections of your code will interact. Take out a piece of paper and pencil and sketch out the major components of your design (collect input, validate input, perform calculations, present results, etc.). Draw simple flow diagrams or write code-like snippets (called *pseudocode*) that capture the major steps in your program. Your design sketch and / or pseudocode is a required deliverable for this assignment.

b. You must make use of formatted output for this assignment. Your output must be neat, and arranged in clearly defined columns. See the sample program run in *Program Flow*, below.

4. **Error Handling**

a. If the user enters an invalid password file name (one that is not found), print a suitable error message and prompt for the file name again. Keep doing this until you get a valid password file.

b. Repeat step 4a for the specification file. Keep looping and prompting until you get a valid specification file.

## 5. Program Flow

Here is a sample specification file (called `pass1.spc`):

```
14
2
3
2
2
```

And here's a sample run of the program using the specification file:

```
*** Password Validator ***

Password File: pass1.txt
Specification File: pass1.spc

Password   1:    ?7)Hr3Dgec:`RM7JF"F=(M`"Y0m ... Valid
Password   2:     f9:YR"%QWxUNq<tBV?fIBbFP/t ... Invalid
Password   3:        #%1"Swn"r>hN+yYB$/.zQG3 ... Valid
Password   4:           V^g+95<d$Y/kt.?%aeCJ ... Valid
Password   5:     -J-_xNd]sI\kh]ijw/uN,AwqfS ... Invalid
Password   6:                 w^70f2cKc?N#]p ... Valid
Password   7:     aeCP<dSz-zW'WGgEp<Y9Nx^OeR ... Invalid
Password   8:             Y9(=wP_g1m*1'lNvm9 ... Valid
Password   9:             mGv*B#D)60qi[><d5_ ... Valid
Password  10:          IvEAJhtf23@yuPf=]4K95 ... Valid

Total Passwords Processed: 10
Valid Passwords: 7
Invalid Passwords: 3
```

## 6. Testing:

a. Thoroughly testing your program can be a difficult task. For complex programs, there may be hundreds or thousands of possible cases to test. The directory for this project also includes several password (`.txt`) and specification (`.spc`) files you can use to test your code. There will be four sets, with each password / specification pair forming a complete set:

   i. `pass1.txt / pass1.spc`
   ii. `pass2.txt / pass2.spc`

iii. `pass3.txt / pass3.spc`
iv. `pass4.txt / pass4.spc`

b. I recommend conducting runs on the test files using the executable version of the program; then run the test files through your version of the program and compare the results.

7. **Assumptions You May Make:**

a. The specification files you use will always have exactly five lines, and will only contain one integer represented on each line.

b. The length of any password you will have to process will not exceed `30` characters.

c. There will never be more than `999` passwords in any password file your program must handle.

8. **Hints:**

a. Give your variables meaningful, descriptive names. While it's true that you could represent a password using a variable named `p`, it will be much easier to read and debug your code if you use a variable name like `passWord` instead.

b. You may find *Liang*, sections 8.1 and 8.2 (starting on p. 242) useful.

c. `strip()`.

d. When I read data from a text file in Python, what type do I get?

e. The *Assumptions You May Make* section above gives you hints that will be helpful in formatting your output.

f. The directory for this project also includes an executable version and several test files that you can run to get a sense for how your program should operate. You should also use these tools to test your own code. The program is called: `validator`.

Let's assume you're working in your `~/shares/sy201` directory. You can run

`validator` by first copying it to your working directory and then adjusting its permissions with the following commands:

i. `cp ~/repo201/programming/pa04/validator .`
(*the trailing space and period in the command above are important*)
ii. `chmod 755 validator`

You only have to perform steps (i) and (ii) once. After that, you can run the program any time you want by typing: `./ validator`

9. **Deliverables And Due Dates:**

a. Using paper, and pen or pencil, complete your pseudocode / flow diagram *before* you start coding. You may use more than one page for this, if you need it. Be sure to indicate your name and alpha on this page.

b. Your completed source code (the one ending in `.py`).

c. Submit parts (a) and (b) by 2359 on Tuesday, 02 October, in accordance with your instructor's directions.