A453: Task 2 System password.

1. Requirements

Design, code test and evaluate a system to accept and test a password for certain characteristics.

- It should be at least 6, and no more than 12 characters long
- The system must indicate that the password has failed and why, asking the user to re enter their choice until a successful password is entered.
- A message to indicate that the password is acceptable must be displayed.
- Password strength can be assessed against simple criteria to assess its suitability; for example a password system using only upper and lower case alphabetical characters and numeric characters could assess the password strength as:

    - WEAK if only one type used, e.g. all lower case or all numeric
    - MEDIUM if two types are used
    - STRONG if all three types are used.

    For example

    - hilltop, 123471324, HAHGFD are all WEAK,
    - catman3 and 123456t are MEDIUM and
    - RTH34gd is STRONG

- A message to indicate the password strength should be displayed after an acceptable password is chosen.

2. Design

It is assumed that the user can run this program from the command line, and enter the password string using the keyboard. Although not specified, the program should also handle error cases such as an empty string, or one that does not contain any letters (lower or uppercase) or digits.

The variables will be:

- password_string – a string input from user. No verification required apart from testing the length of the string at the start of the program.
- lowercase_found – integer (0 or 1)
- uppercase_found – integer (0 or 1)
- numeric_found – integer (0 or 1)
- password_strength – integer (sum of lowercase_found, uppercase_found and numeric_found)

The result provided to the user should test the value of password_strength and report accordingly. If it is still zero, then an error should be reported as neither lowercase, uppercase or digits were found when the string was examined.

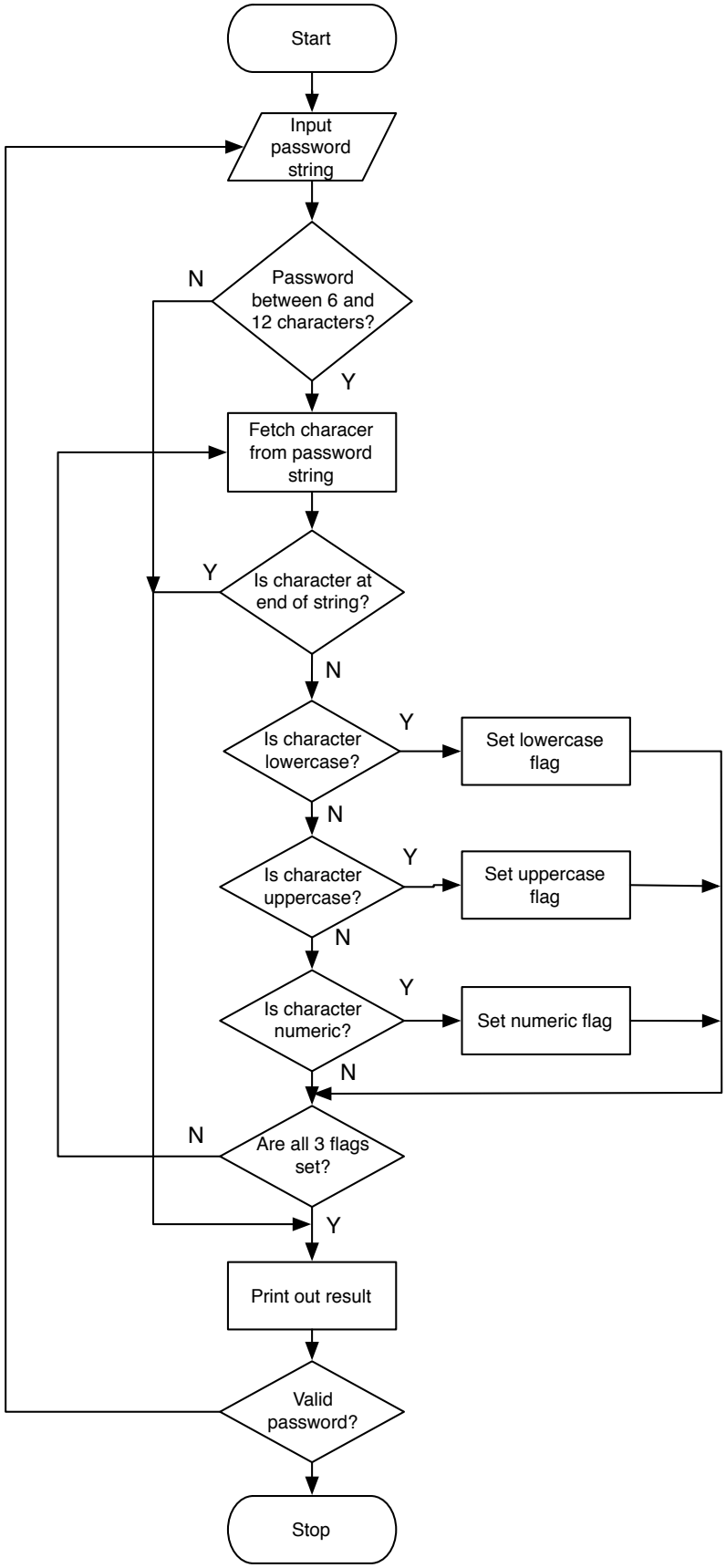
Two design approaches are given below, firstly in pseudo code, and secondly with a flowchart.

2.1 Pseudo code

The pseudo code is very straightforward, and translates easily into the Python implemented as listed in the development stages in section 3.

```
While valid password not entered
    Input password string
    If password string less than 6 characters long, then print out message and exit
    If password string is more than 12 characters long, then print out message and
    exit
    Set lowercase, uppercase and numerical flags to 0
    Step through each character in string
        If character is lowercase, then set lowercase flag
        If character is uppercase, then set uppercase flag
        If character is numeric, then set numerical flag
        If all three flags set, exit loop early
    Add three flags to form password_strength
    If password_strength is between 1 and 3 inclusive
        Print out message that password is valid
        If password_strength is 1, then print out WEAK password message
        If password_strength is 2, then print out MEDIUM password message
        If password_strength is 3, then print out STRONG password message
    Else print out error message
```

2.2 Flowchart



3. Development

3.1 Development choices

In developing an appropriate solution to the task, I used the Python programming language, due to its familiarity to the author, and also it availability and power in implementing such tasks. The Python 'isupper()', 'islower()' and 'isdigit()' built-in functions were used, rather than implementing these by hand (which may be error-prone and less efficient) instead testing the ordinal numbers of each character along the string (e.g. using the 'ord()' function).

The features of the language utilised are conditional statements are as follows:

1. The 'while' loop should be used to ensure the user keeps entering a password until it has been evaluated as valid (i.e. between 6 and 12 characters long, inclusive, and containing lowercase, uppercase or digits).
2. The Python 2 'raw_input()' function was used to fetch the password string from the command line.
3. Conditional 'if' statements that test the length of the string, and whether the character being examined is lowercase, uppercase, or a digit.
4. An iterative 'for' loop is used to step over the password string, one character at a time. A 'while' loop was not used as this requires more code in setting up a control variable, and incrementing this on a per loop basis. Therefore, a 'for' loop was deemed more appropriate.
5. The 'break' command was used to efficiently break out of the loop early when all the appropriate flags have been set
6. The 'print' command was used to print out the result.

Comments have been used to annotate the code where appropriate. No functions have been used with this version of the program – it is presented as a Python 'script' that can be run quickly as easily. A more modularised version with functions (as required) is presented in the automated unit testing in section 4.2.

3.2 Test plan

The overall test plan is as follows. This allows the program to be tested as it is developed, as well as the final solution, particularly the boundary cases. Although the requirements do not mention an empty password, this is particularly likely to trip up a program if not caught. Other boundary cases are for the password to be between 6 and 12 characters, and the combination of invalid, WEAK, MEDIUM and STRONG passwords.

Columns in the table are as follows:

- number so we can refer to them specifically later
- the purpose behind the test, as each test must fulfil path through the code
- the input required
- the expected outcome

- the actual outcome from running the program code
- whether the program passed the test or not (✓ or ✗).

Obviously, the final two columns cannot be filled out (therefore, they are filled with To Be Confirmed - TBC) until the program has been developed and tested.

No.	Purpose	Input password	Expected outcome	Actual outcome	✓ / ✗
1	Check whether program can handle empty password	''	Error – too short Ask for password again	TBC	TBC
2	Test password that is too short (5 chars)	abcde	Error – too short Ask for password again	TBC	TBC
3	Test password that is too long (13 chars)	abcdefghijklm	Error – too long Ask for password again	TBC	TBC
4	Test password that is invalid	!@£$%^&*()	Error – invalid Ask for password again	TBC	TBC
5	Test minimum of 6 characters – lowercase only	abcdef	WEAK password	TBC	TBC
6	Test minimum of 6 characters – uppercase only	ABCDEF	WEAK password	TBC	TBC
7	Test minimum of 6 characters – digits only	123456	WEAK password	TBC	TBC
8	Test maximum of 12 characters	123456789012	WEAK password	TBC	TBC
9	Test combination of lowercase and uppercase	aBcDeF	MEDIUM password	TBC	TBC
10	Tests combination of lowercase and digits	abc123	MEDIUM password	TBC	TBC
11	Test combination of uppercase and digits	ABC123	MEDIUM password	TBC	TBC
12	Test combination of lowercase, uppercase and digits	aBc1Ef	STRONG password	TBC	TBC

3.3 Stage 1 – User input and validation

The first stage of development is to accept the password string from the user and test whether it is within the range deemed as acceptable, i.e. between 6 and 12 characters, inclusive.

The program code for this stage is as follows:

```
# Input password string from the user with an appropriate question
password_str = raw_input('Please enter password to test: ')

# Test if password_str is the correct length, and skip loop if not
if len(password_str) < 6:
    print 'Password:', password_str, 'too short!  It must be between 6 and 12 characters.'
elif len(password_str) > 12:
    print 'Password:', password_str, 'too long!  It must be between 6 and 12 characters.'
else:
    print 'Password is the correct length!'
```

The testing that can now be tested so far is as follows:

No.	Purpose	Input password	Expected outcome	Actual outcome	✓ / ✗
1	Check whether program can handle empty password	''	Error – too short Ask for password again	Error – too short Does not ask for password again	✗
2	Test password that is too short (5 chars)	abcde	Error – too short Ask for password again	Error – too short Does not ask for password again	✗
3	Test password that is too long (13 chars)	abcdefghijklm	Error – too long Ask for password again	Error – too long Does not ask for password again	✗

The program was run to provide the results above as follows:

```
joyce-book:python joyce$
joyce-book:python joyce$ python password_test.py
Please enter password to test:
Password:  too short!  It must be between 6 and 12 characters.
joyce-book:python joyce$ python password_test.py
Please enter password to test: abcde
Password: abcde too short!  It must be between 6 and 12 characters.
joyce-book:python joyce$ python password_test.py
Please enter password to test: abcdefghijklm
Password: abcdefghijklm too long!  It must be between 6 and 12 characters.
joyce-book:python joyce$
```

The program fails to ask for the password again when it reports an error, but this will be added later. With this stage successfully complete, we can progress onto the next stage.

3.4 Stage 2 – stepping through the password string

The above program will now be expanded so that the three flags are initially set to zero (lowercase_found, uppercase_found and numeric_found), and then the password is examined one character at a time using the Python 'for' loop. If an appropriate character is found (using the built-in string methods is_lower(), is_upper() and is_digit()), then the flags will be set to 1 as needed. A check as to whether these flags are already set is not required, as simply setting them is both simple and efficient. However, it is efficient to break out of the loop early if all three flags are set, as continuing will not add any further information. Finally, the flags are combined together, and the appropriate message is output.

```
# Input password string from the user with an appropriate question
password_str = raw_input('Please enter password to test: ')

# Test if password_str is the correct length, and skip loop if not
if len(password_str) < 6:
    print 'Password:', password_str, 'too short!  It must be between 6 and 12 characters.'
elif len(password_str) > 12:
    print 'Password:', password_str, 'too long!  It must be between 6 and 12 characters.'
else:
    # Set flags to be set when different tests succeed
    lowercase_found = 0
    uppercase_found = 0
    numeric_found = 0

    # Step through string and set flags where appropriate
    for ch in password_str:
        if ch.islower():
            lowercase_found = 1
        if ch.isupper():
            uppercase_found = 1
        if ch.isdigit():
            numeric_found = 1

        # Exit early if all our flags have been set
        if lowercase_found and uppercase_found and numeric_found:
            break

    # Calculate password strength, and print out result
    password_strength = lowercase_found + uppercase_found + numeric_found

    # If the password is either 1, 2 or 3, then print out valid message.
    if password_strength in range(1,4):
        print 'The password', password_str, 'is valid.'

        # Now print out password strength
        if password_strength == 1:
            print 'Password', password_str, 'is a WEAK password'
        elif password_strength == 2:
            print 'Password', password_str, 'is a MEDIUM password'
        else:
            print 'Password', password_str, 'is a STRONG password'
    else:
        print 'Invalid password:', password_str, 'contained neither letters or numbers!'
```

The testing that can now be fulfilled is as follows:

No.	Purpose	Input password	Expected outcome	Actual outcome	✓ / ✗
1	Check whether program can handle empty password	''	Error – too short Ask for password again	Error – too short Does not ask for password again	✗
2	Test password that is too short (5 chars)	abcde	Error – too short Ask for password again	Error – too short Does not ask for password again	✗
3	Test password that is too long (13 chars)	abcdefghijklm	Error – too long Ask for password again	Error – too long Does not ask for password again	✗
4	Test password that is invalid	!@£$%^&*()	Error – invalid Ask for password again	Error – invalid Does not ask for password again	✗
5	Test minimum of 6 characters – lowercase only	abcdef	WEAK password	WEAK password	✓
6	Test minimum of 6 characters – uppercase only	ABCDEF	WEAK password	WEAK password	✓
7	Test minimum of 6 characters – digits only	123456	WEAK password	WEAK password	✓
8	Test maximum of	123456789012	WEAK password	WEAK password	✓

	12 characters				
9	Test combination of lowercase and uppercase	aBcDeF	MEDIUM password	MEDIUM password	✓
10	Tests combination of lowercase and digits	abc123	MEDIUM password	MEDIUM password	✓
11	Test combination of uppercase and digits	ABC123	MEDIUM password	MEDIUM password	✓
12	Test combination of lowercase, uppercase and digits	aBc1Ef	STRONG password	STRONG password	✓

The first four test cases still fail, but this is to be expected as the password re-entry loop has not been implemented.

The program was run to provide the results above as follows:

```
joyce-book:python joyce$ python password_test.py
Please enter password to test:
Password:  too short!  It must be between 6 and 12 characters.
joyce-book:python joyce$ python password_test.py
Please enter password to test: abcde
Password: abcde too short!  It must be between 6 and 12 characters.
joyce-book:python joyce$ python password_test.py
Please enter password to test: abcdefghijklm
Password: abcdefghijklm too long!  It must be between 6 and 12 characters.
joyce-book:python joyce$ python password_test.py
Please enter password to test: !@£$%^&*()
Invalid password: !@£$%^&*() contained neither letters or numbers!
joyce-book:python joyce$ python password_test.py
Please enter password to test: abcdef
The password abcdef is valid.
Password abcdef is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: ABCDEF
The password ABCDEF is valid.
Password ABCDEF is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: 123456
The password 123456 is valid.
Password 123456 is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: 123456789012
The password 123456789012 is valid.
Password 123456789012 is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: aBcDeF
The password aBcDeF is valid.
Password aBcDeF is a MEDIUM password
joyce-book:python joyce$ python password_test.py
Please enter password to test: abc123
The password abc123 is valid.
Password abc123 is a MEDIUM password
joyce-book:python joyce$ python password_test.py
Please enter password to test: ABC123
The password ABC123 is valid.
Password ABC123 is a MEDIUM password
joyce-book:python joyce$ python password_test.py
Please enter password to test: aBc1Ef
The password aBc1Ef is valid.
Password aBc1Ef is a STRONG password
joyce-book:python joyce$
```

Therefore, the program is almost complete, but lacks an overall loop which will require the user to re-enter the password if it is found to be invalid.

3.5 Stage 3 (final stage) – add loop for invalid passwords

The final stage in the development of this task is to add a loop so that the user is required to re-enter the password if it is found to be invalid. This simply sets a Boolean flag at the start of the program, which is set if the criteria (between 6 and 12 characters, containing at least one lowercase, uppercase or digit character) is satisfied.

The final program listing is as follows:

```
# Set flag so that user must enter valid password
valid_password_entered = False

# Loop until valid password is entered
while not valid_password_entered:
    # Input password string from the user with an appropriate question
    password_str = raw_input('Please enter password to test: ')

    # Test if password_str is the correct length, and skip loop if not
    if len(password_str) < 6:
        print 'Password:', password_str, 'too short!  It must be between 6 and 12 characters.'
    elif len(password_str) > 12:
        print 'Password:', password_str, 'too long!  It must be between 6 and 12 characters.'
    else:
        # Set flags to be set when different tests succeed
        lowercase_found = 0
        uppercase_found = 0
        numeric_found = 0

        # Step through string and set flags where appropriate
        for ch in password_str:
            if ch.islower():
                lowercase_found = 1
            if ch.isupper():
                uppercase_found = 1
            if ch.isdigit():
                numeric_found = 1

            # Exit early if all our flags have been set
            if lowercase_found and uppercase_found and numeric_found:
                break

        # Calculate password strength, and print out result
        password_strength = lowercase_found + uppercase_found + numeric_found

        # If the password is either 1, 2 or 3, then print out valid message.
        if password_strength in range(1,4):
            print 'The password', password_str, 'is valid.'

            # Now print out password strength
            if password_strength == 1:
                print 'Password', password_str, 'is a WEAK password'
            elif password_strength == 2:
                print 'Password', password_str, 'is a MEDIUM password'
            else:
                print 'Password', password_str, 'is a STRONG password'

            # Set loop variable, so that we do not ask for password again
            valid_password_entered = True
        else:
            print 'Invalid password:', password_str, 'contained neither letters or numbers!'
```

The test plan can now be re-run with the results from the version above as follows:

No.	Purpose	Input password	Expected outcome	Actual outcome	✓ / ✗
1	Check whether program can handle empty password	''	Error – too short Ask for password again	Error – too short Ask for password again	✓
2	Test password that is too short (5 chars)	abcde	Error – too short Ask for password again	Error – too short Ask for password again	✓
3	Test password that is too long (13 chars)	abcdefghijklm	Error – too long Ask for password again	Error – too long Ask for password again	✓
4	Test password that is invalid	!@£$%^&*()	Error – invalid Ask for password again	Error – invalid Ask for password again	✓
5	Test minimum of	abcdef	WEAK password	WEAK password	✓

	6 characters – lowercase only				
6	Test minimum of 6 characters – uppercase only	ABCDEF	WEAK password	WEAK password	✓
7	Test minimum of 6 characters – digits only	123456	WEAK password	WEAK password	✓
8	Test maximum of 12 characters	123456789012	WEAK password	WEAK password	✓
9	Test combination of lowercase and uppercase	aBcDeF	MEDIUM password	MEDIUM password	✓
10	Tests combination of lowercase and digits	abc123	MEDIUM password	MEDIUM password	✓
11	Test combination of uppercase and digits	ABC123	MEDIUM password	MEDIUM password	✓
12	Test combination of lowercase, uppercase and digits	aBc1Ef	STRONG password	STRONG password	✓

Evidence of manual test plan by running program repeatedly on the command line:

```
joyce-book:python joyce$ python password_test.py
Please enter password to test:
Password:  too short!  It must be between 6 and 12 characters.
Please enter password to test: abcde
Password: abcde too short!  It must be between 6 and 12 characters.
Please enter password to test: abcdefghijklm
Password: abcdefghijklm too long!  It must be between 6 and 12 characters.
Please enter password to test: !@£$%^&*()
Invalid password: !@£$%^&*() contained neither letters or numbers!
Please enter password to test: abcdef
The password abcdef is valid.
Password abcdef is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: ABCDEF
The password ABCDEF is valid.
Password ABCDEF is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: 123456
The password 123456 is valid.
Password 123456 is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: 123456789012
The password 123456789012 is valid.
Password 123456789012 is a WEAK password
joyce-book:python joyce$ python password_test.py
Please enter password to test: aBcDeF
The password aBcDeF is valid.
Password aBcDeF is a MEDIUM password
joyce-book:python joyce$ python password_test.py
Please enter password to test: abc123
The password abc123 is valid.
Password abc123 is a MEDIUM password
joyce-book:python joyce$ python password_test.py
Please enter password to test: ABC123
The password ABC123 is valid.
Password ABC123 is a MEDIUM password
joyce-book:python joyce$ python password_test.py
Please enter password to test: aBc1Ef
The password aBc1Ef is valid.
Password aBc1Ef is a STRONG password
joyce-book:python joyce$
```

The conclusion of manual testing is that the program works according to the original requirements laid out in section 1, but a more thorough evaluation is provided in the next section.

4. Evaluation of the solution

The program has been designed according to the problem requirements, and given thorough manual testing. Test plans have been filled out as the program development has progressed. Screenshots of the program being run have been captured to demonstrate the test cases being successfully completed. All the tests, which concentrated on boundary cases according to the requirements, have passed.

Now we will revisit the requirements and match the test plan results against each item to ensure we have fulfilled them properly:

- It should be at least 6, and no more than 12 characters long
    - this is satisfied by test cases 1, 2, 3 and 8 which passed.
- The system must indicate that the password has failed and why, asking the user to re enter their choice until a successful password is entered.
    - this is satisfied by test cases 1, 2 and 3 which passed
- A message to indicate that the password is acceptable must be displayed.
    - this is satisfied by test cases 5 to 12 inclusive, which prints out a valid message for those passwords that are either WEAK, MEDIUM or STRONG.
- Password strength can be assessed against simple criteria to assess its suitability; for example a password system using only upper and lower case alphabetical characters and numeric characters could assess the password strength as:
    - WEAK if only one type used, e.g. all lower case or all numeric
    - MEDIUM if two types are used
    - STRONG if all three types are used.
    - this is satisfied by test cases 5 to 12 (similar to the next case which prints out the result)
- A message to indicate the password strength should be displayed after an acceptable password is chosen.
    - this is satisfied by test cases 5 to 12 inclusive, varying between WEAK, MEDIUM and STRONG passwords.

Therefore, combined with the test plan results, the task has been designed and developed to satisfy all the requirements specified in the task.