2020L219A2EL

Coimisiún na Scrúduithe Stáit
State Examinations Commission

Leaving Certificate Examination 2020

Computer Science

Section C

Higher Level

1 hour

80 marks

Instructions

There is one section in this paper.

Section C	Programming	80 marks	1 question

Answer all parts of the question on your digital device.

Calculators may be used during this section of the examination.

The *Formulae and Tables* booklet cannot be used for this section of the examination.

The Superintendent will give you a copy of the *Python Reference Guide*.

Ensure that you save your work regularly and when you complete each question part.

Save your files using the naming structure described at the beginning of each question part.

If you are unable to get some code to work correctly, you can comment out the code so that you can proceed. The code that has been commented out will be reviewed by the examiner.

Rough work pages are provided at the end of this booklet. Please note that this booklet is not to be handed up and will **not** be reviewed by an examiner.

At the end of the examination it is your responsibility to ensure that you have saved all of your files onto your external media.

You will be provided with a brown envelope for your external media. Write your examination number on this envelope and place your external media into it before sealing. Place this envelope in the pouch at the front of the red envelope that contains your examination booklet from Section A and B.

Do not hand this paper up

Section C — Programming — 80 marks

Answer all question parts.

Question 16

A password strength meter is a mechanism that can be used to safeguard against setting weak passwords. When a user is creating a password for the first time or changing an existing password, a password strength meter can be used to show how resistant the password is to attack.

Meters have rules they use to assign points for password strengthening measures such as including combinations of uppercase and lowercase letters as well as numbers and special symbols.

(a) Open the program called **Question16_A.py** from your device. The source code is shown on the next page and described briefly below.

Before making any changes, you should use the format **CandidateNumberQuestion16_A.py** to save your file. For example, if your candidate number was 123456 you would save the file as **123456Question16_A.py**.

Enter your Examination Number in the space provided on **line 2.**

This program is designed to calculate and display a score that indicates the strength of a password entered by the user.

The variable **`score`** is used to store the password strength. This variable is initially set to zero and additional points are added based on the following rules:

1. The password contains more than seven characters: $+5$ points
2. The password contains at least one lowercase letter: $+1$ point
3. The password contains a mix of lowercase and uppercase letters. $+5$ points

A sample run of the program is shown below:

```
Enter a password: sunshine
6
```

Here the user enters the password *sunshine* and the program calculates and displays a score of 6. This is because the password contains more than seven characters (5 points) and contains lowercase letters (1 point).

```
# Question 16(a)
# Examination Number:

# Prompt the user to enter a password and store the ...
# value entered in the variable password
password = input("Enter a password: ")

# A variable to store all the lowercase letters in the alphabet
LOWER_CASE_LETTERS = "abcdefghijklmnopqrstuvwxyz"

# The variables lowercase and uppercase indicate the presence or ...
# absence of lowercase and uppercase characters in the password
lowercase = False # True if password contains a lowercase letter
uppercase = False # True if password contains an uppercase letter

# Loop through each character in the password and ...
# check the password for specific characters
for character in password:
  if character in LOWER_CASE_LETTERS:
    lowercase = True
  if character in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
    uppercase = True

# Calculate the score based on the rules

score = 0

# Rule 1
if len(password) > 7:
    score = score + 5

# Rule 2
if lowercase:
    score = score + 1

# Rule 3
if lowercase and uppercase:
    score = score + 5

# Display the score
print(score)

```

Make the following changes to the program:

(i) Insert a comment to say '*initialise score'* in an appropriate location in the program.

(ii) Amend the program so that is displays two lines of output as follows:

- the first line will display the word *Password:* followed by the password that was entered by the user, and
- the second line will display the word *Score:* followed by the calculated score for that password.

When the program is run the output may look as follows:

```
Enter a password: sunshine
Password: sunshine
Score: 6
```

(iii) Currently in the program, the uppercase letters are hard-coded as the string: `"ABCDEFGHIJKLMNOPQRSTUVWXYZ"`

Replace the use of this string with a variable, in a manner similar to that used to represent the lowercase letters. The output of the program should not be changed. You should name the variable **`UPPER_CASE_LETTERS`**.

(iv) Implement a new rule (rule 4) so that the score is increased by 2 points if the password contains at least one uppercase letter.

When the program is run the output may look as follows:

```
Enter a password: Sunshine
Password: Sunshine
Score: 13
```

(v) Implement a new rule (rule 5) so that the score is increased by 5 points if the password contains at least one digit (any integer in the range 0 to 9: $+5$ points).

When the program is run the output may look as follows:

```
Enter a password: 3Sunshine
Password: 3Sunshine
Score: 18
```

(vi) Implement a new rule (rule 6) so that the score is increased by:

- 1 point if the first character of the password is a digit
- 1 point if the last character of the password is a digit
- 2 extra points if both the first and the last characters of the password are digits

When the program is run the output may look as follows:

```
Enter a password: 3Sunshine7
Password: 3Sunshine7
Score: 22
```

(vii) Implement a new rule (rule 7) so that the score is reduced by 10 points if the password contains only digits.

When the program is run the output may look as follows:

```
Enter a password: 1234
Password: 1234
Score: -1
```

(viii) Change rule 1 so that the score is adjusted according to the password lengths as shown in the following table.

Password Length	**Score**
Greater than 7 characters	$+5$ points
From 4 to 7 characters	$+2$ points
Less than 4 characters	-2 points

The table below shows the scores that would be awarded for a variety of passwords. You could use this information to test your program.

Password	**Score**
sun	-1
Sun	6
sun2	9
2sun3	12
3Sunshine	19
3Sunshine7	22

Use the format **CandidateNumberQuestion16_A.py** to save your file. For example, if your candidate number was 123456 you would save the file as **123456Question16_A.py**.

(b) Open the program called **Question16_B.py** from your device. The source code is shown on the next page and described briefly below.

Before making any changes, you should use the format **CandidateNumberQuestion16_B.py** to save your file. For example, if your candidate number was 123456 you would save the file as **123456Question16_B.py**.

Enter your Examination Number in the space provided on l**ine 2.**

This program is very similar to that provided for part (a) with two main differences:

- The code to calculate the password score is contained in a function definition called **`calculate_score`**. This function accepts a parameter called **`password`** and returns the calculated score.
- Instead of prompting the user to enter a single password this program, uses a list of hard-coded passwords called **`test_passwords`**.

When the program is run it loops through each password in the list **`test_passwords`**. As it does so, it calculates and displays the score of each password.

A sample run of the program is shown below:

```
1
6
11
0
5
```

```python
# Question 16(b)
# Examination Number:

# A variable to store all the lower case letters in the alphabet
LOWER_CASE_LETTERS = "abcdefghijklmnopqrstuvwxyz"

def calculate_score(password):

    # The variables lowercase and uppercase indicate the presence or
    # absence of lowercase and uppercase characters in the password
    lowercase = False #True if password contains a lowercase letter
    uppercase = False #True if password contains an uppercase letter

    # Loop through each character in the password and ...
    # ... check the password for specific characters
    for character in password:
      if character in LOWER_CASE_LETTERS:
        lowercase = True
      if character in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
        uppercase = True

    # Calculate the score based on the rules

    score = 0

    # Rule 1
    if len(password) > 7:
        score = score + 5

    # Rule 2
    if lowercase:
        score = score + 1

    # Rule 3
    if lowercase and uppercase:
        score = score + 5

    return score

# Test driver
test_passwords = ["sun", "Sun", "Sunshine", "12345", "123456789"]
for password in test_passwords:
    pass_score = calculate_score(password)
    print(pass_score)
```

Make the following changes to the program:

(i) Amend the program so that the output is displayed in the following format:

```
Score   Password
-----   --------
1       sun
6       Sun
11      Sunshine
0       12345
5       123456789
```

(ii) Insert a line of code to change the password contained at index 4 of the list **`test_passwords`** from *123456789* to *Moonlight*.

When the program is run the output may look as follows:

```
Score   Password
-----   --------
1       sun
6       Sun
11      Sunshine
0       12345
11      Moonlight
```

(iii) Amend the program so that it determines and displays the weakest password in the list along with its score.

When the program is run the output may look as follows:

```
Score   Password
-----   --------
1       sun
6       Sun
11      Sunshine
0       12345
11      Moonlight

The weakest password is: 12345
Score: 0
```

(iv) Write a function definition called **`is_strong`** which accepts a password as a parameter and returns **`True`** if the password is strong; **`False`** otherwise.

A password is deemed strong if it contains more than seven characters and both lowercase and uppercase letters.

The first line of the function definition will look like this:

```
def is_strong(password):
```

(v) Modify the program so that it calls the function **`is_strong`** for each password in the list **`test_passwords`** and displays all the strong passwords.

When the program is run the output may look as follows:

```
Score   Password
-----   --------
1       sun
6       Sun
11      Sunshine
0       12345
11      Moonlight

The weakest password is: 12345
Score: 0

The strong passwords are:
Sunshine
Moonlight
```

Use the format **CandidateNumberQuestion16_B.py** to save your file. For example, if your candidate number was 123456 you would save the file as **123456Question16_B.py**.

Space for rough work.

This page will not be reviewed by an examiner.

Space for rough work.

This page will not be reviewed by an examiner.

Space for rough work.

This page will not be reviewed by an examiner.

Space for rough work.

This page will not be reviewed by an examiner.

Space for rough work.

This page will not be reviewed by an examiner.

Leaving Certificate – Higher Level

Computer Science – Section C

1 hour