Simon Fraser University

CMPT 120 D300

Spring 2024

Friday March 8 – 12:30pm

Midterm - **SOLUTION**

Time: 45 minutes – out of 45 marks

This examination has 12 pages.

Read each question carefully before answering it.

- No textbooks, calculators, computers, cell phones or other materials may be used.
- You must use **Python 3** syntax.
- **Always comment your code** and always use the **Good Programming Style** (GPS) discussed in our lectures.
- Questions are not permitted during this midterm. However, you can write down any assumptions you make when answering a question.
- Hand in your **cheat sheet** along with your examination paper. **Do not put** your cheat sheet inside your examination paper when you hand them in.
- The marks for each question are given in [ ]. Use this to manage your time.

Good luck!

Part 1 - Theory and Understanding - [11 marks in total – 1 mark per question - No part marks] Answer the following multiple-choice questions on the **bubble sheet** at the back of this examination paper. Select only one (1) answer per question.

1. Consider the Python fragment below:

```
gameOver = False
theAnswer = 6 - 4 + 8 * 10

_________________________
    print(f"The answer is {theAnswer}.")
```

Which of the following Python statement would create an error if it were used to fill in the blank line above?

A. `if gameOver :`
B. `if theAnswer == 100 :`
C. `if theAnswer.isdigit( ) :`
D. All of the above
E. None of the above

2. What can we say about the following two Python code fragments?

Python Code Fragment A

```
age = int(input("Age: "))
if age < 15 :
   print("drink milk")
if age < 18 :
   print("drink coffee")
else:
   print("drink ???")
```

Python Code Fragment B

```
age = int(input("Age: "))
if age < 15 :
   print("drink milk")
elif age < 18 :
   print("drink coffee")
else:
   print("drink ???")
```

A. Python code fragment A produces an error.
B. They both produce the same result for any input the user enters.
C. They do not always produce the same result.
D. Python code fragment B produces an error.
E. None of the above

3. What does the following Python code fragment print on the computer monitor screen?

```
theLength = len("awesome")
theList = [0, "number", theLength, len("awesome"), 0]
print(theList[5])
```

A. `[0, "number", theLength, len("awesome"), 0]`

B. There is an error in the code: a list cannot have the same element twice (here `0`).

C. `0`

D. `5`

E. None of the above

4. What would the variable `colours` contain at the end of the 2nd iteration of the following Python loop:

```
colours = ["white", "green", "red", "yellow"]
for colour in colours :
    print(colour)
```

A. Python loop produces an error.

B. `yellow`

C. `["yellow"]`

D. `["white", "green", "red", "yellow"]`

E. None of the above

5. What does the following Python code fragment print on the computer monitor screen?

```
import random
numberOfIdeas = random.randint(1, 10)
if numberOfIdeas == 1:
   print("You only have 1 idea!")
else:
   print("You are full of ideas!")
```

A. There are no errors in the code, but we cannot tell what the result will be.

B. `You are full of ideas!`

C. There is an error in the code.

D. `You only have 1 idea!`

E. None of the above

6. What does the following Python code fragment print on the computer monitor screen?

```
def pow(base, exponent):
  result = base**exponent
  return result

def square(operand):
  return pow(operand, 2)

# ***Main part of the program
aNumber = 5
result = square(aNumber)
print(result + 15)
```

A. `40`

B. It produces an error: the function `square(…)` should be defined before the function `power(…)`.

C. `2515`

D. `25`

E. None of the above

7. To what value should we initialize a variable (i.e., assign an initial value to the variable before using it) if we want to use this variable as a running sum (accumulator) variable in our program?

A. Set the variable to any number.

B. Set the variable to 1.

C. Set the variable to an empty string.

D. If the variable is to be used as a running sum variable, it should not be initialized at all.

E. None of the above

8. What would the variable `slice` contain once the following Python code fragment has executed?

```
prov = "British Columbia"
slice = prov.upper()[3:13:2]
```

A. `"ts ou"`

B. `"TS OU"`

C. `"TS OUB"`

D. There is an error in the code.

E. None of the above

9. Which of the following is NOT true?

   A. The statements in the body of a `for` loop must be indented.

   B. `while` loops iterate as long as their condition evaluates to `True`.

   C. We use a `for` loop when we do not know how many times the statements in the body of the loop need to be executed.

   D. Variables declared in a function have local scope: they no longer exist once the function has stopped executing.

   E. None of the above

10. What does `print((16 - 6 // 2 * 3) % 3 )` print on the computer monitor screen?

    A. `7`

    B. `1`

    C. `5`

    D. There is an error in the code.

    E. None of the above

11. What does the following Python code fragment print on the computer monitor screen?

```
string = "WPLM"
ending = "ow"
for letter in string:
   print(f'{letter + ending})
```

    A. `Wow`
       `Pow`
       `Low`
       `Mow`

    B. `Wow Pow Low Mow`

    C. `WPLMow`

    D. There is an error in the code.

    E. None of the above

12. [2 marks – No part marks] Write the header of a Python function which computes the perimeter of a rectangle given its two sides. Warning: Do not write more than what is asked for.

    Possible answer:

```
def rectanglePerimeter(aSide1, aSide2):
```

Part 2 – Coding - The weight of each question is indicated in [ ] – When writing your answers, do not write over the QR symbol at the top of the pages.

1. [12 marks]

   Step 1 - Problem Statement

   On the next two pages, you will find an incomplete **Encryption** program. It asks the user to enter a message to encrypt, it calls the encrypt function then prints the message and the corresponding encrypted message. But the encrypt function is missing. Your task is to complete this program by implementing the encrypt function in the given space on the next two pages.

   Step 2 – Design

   In order for you to know which encryption algorithm you need to implement when completing this **Encryption** program, have a look at the following table. It contains examples of plain messages and their corresponding cipher (encrypted) messages. Looking at these examples, can you guess the encryption algorithm you must implement when developing the encrypt function?

Plain Messages	Corresponding Cipher Messages
Hello! How are you doing?	H2ll4! H4w 1r2 y45 d43ng?
Sorry, I can't talk right now. I am writing a midterm.	S4rry, 3 c1n't t1lk r3ght n4w. 3 1m wr3t3ng 1 m3dt2rm.
Oh! Can I help you?	4h! C1n 3 h2lp y45?
NO, YOU CANNOT!	N4, Y45 C1NN4T!
Oops! Sorry I disturbed you! I shall leave you in peace now! Good luck!	44ps! S4rry 3 d3st5rb2d y45! 3 sh1ll l21v2 y45 3n p21c2 n4w! G44d l5ck!

   You can assume that the user is well-behaved: they will always enter what you are asking for. If you ask for a message (i.e., a string), the user will enter a message (i.e., a string).

   Note that you are not told which Python statements to use in order to implement this function. You need to use your own judgment and your software development experience acquired so far in this course to make the proper decisions.

Part 2 - Question 1 – Encryption Program

```
# ***Header comment block***
# Imagine this is a complete header comment block
# You do not have to add anything to it!
```

Possible solutions:

```
# Encrypt.py
#
# Midterm - Part 2 - Question 1
#
# Description: Encryption program where
#              a -> 1, e -> 2, i -> 3, o -> 4 and u -> 5
# Author: AL
# Date: Feb. 2024


def encrypt_v1(aPlainMsg):
    """Encrypt plainMsg using this algorithm:
       a -> 1, e -> 2, i -> 3, o -> 4 and u -> 5."""

    # Initialize the accumulator
    aCipherMsg = ""

    # For each char in the message:
    for char in aPlainMsg :
    # *** Note that this is considered "repeated code" ***
        if char == 'a' or char == 'A': aCipherMsg += '1'
        elif char == 'e' or char == 'E': aCipherMsg += '2'
        elif char == 'i' or char == 'I': aCipherMsg += '3'
        elif char == 'o' or char == 'O': aCipherMsg += '4'
        elif char == 'u' or char == 'U': aCipherMsg += '5'
        else : aCipherMsg += char
    # Is there a better way to solve this problem? See v2 below

    # Return encrypted message (cipher)
    return aCipherMsg
```

Part 2 - Question 1 (cont'd)

```
def encrypt_v2(aPlainMsg):
    """Encrypt plainMsg using this algorithm:
       a -> 1, e -> 2, i -> 3, o -> 4 and u -> 5."""

    # Initialize the local variables
    vowels = "aeiou"
    encriptChars = "12345"

    # Initialize the accumulator
    aCipherMsg = ""

    # For each char in the message:
    # Do you see the difference between v1 above and v2
    for char in aPlainMsg :
        if char.lower() in vowels :
            theIndex = vowels.index(char.lower())
            # Accumulate the cipher with encryted vowel
            aCipherMsg += encriptChars[theIndex]
        else :
            # Accumulate the cipher with non-vowel character
            aCipherMsg += char

    # Return encrypted message (cipher)
    return aCipherMsg
```

```python
# How about this one?
def encrypt_v3(aPlainMsg):
    """Encrypt plainMsg using this algorithm:
       a -> 1, e -> 2, i -> 3, o -> 4 and u -> 5."""

    # Using a dictionary
    vowelDict = { "a":"1", "e":"2", "i":"3", "o":"4", "u": "5",
                  "A":"1", "E":"2", "I":"3", "O":"4", "U": "5" }

    # Initialize the accumulator
    aCipherMsg = ""

    # For each char in the message :
    for char in aPlainMsg :
        # If the key is found in the dictionary (key being a vowel),
        # the function get(...) returns the value associated
        # with the key, i.e., its value (its encrypted vowel)
        # If the key is not found in the dictionary (key being a non-vowel),
        # the function get(...) returns its 2nd argument,
        # i.e., "char" (the non-vowel character)
        # Accumulate the cipher with "char"
        # (being the encrypted vowel or the non-vowel char)
        aCipherMsg += vowelDict.get(char,char)

    # Return encrypted message (cipher)
    return aCipherMsg
```

```
# ***Main part of the program

# Ask the user for a message - the user is well-behaved!
plainMsg = input("Please, enter a message to encrypt: ")

# Call the encrypt function
cipherMsg1 = encrypt_v1(plainMsg)

# Print the original message and the encrypted message
print(f"'From version 1: {plainMsg}' becomes '{cipherMsg1}'.")

# Call the encrypt function
cipherMsg2 = encrypt_v2(plainMsg)

# Print the original message and the encrypted message
print(f"'From version 2: {plainMsg}' becomes '{cipherMsg2}'.")

# Call the encrypt function
cipherMsg3 = encrypt_v3(plainMsg)

# Print the original message and the encrypted message
print(f"'From version 3: {plainMsg}' becomes '{cipherMsg3}'.")

# End of program
```

```
# ***Main part of the program

# Ask the user for a message - the user is well-behaved!
plainMsg = input("Please, enter a message to encrypt: ")

# Call the encrypt function
cipherMsg = encrypt(plainMsg)

# Print the original message and the encrypted message
print(f"'{plainMsg}' becomes '{cipherMsg}'.")

# End of program
```

2. [20 marks]

Step 1 - Problem Statement

A vending machine company has asked you to write a **VendingMachine** program, which is to execute on the company's latest model of vending machines. This program is to allow customers to select various items sold in these vending machines and to compute the total cost of the item(s) customers have selected. Examine the samples runs below to determine how this program is to interact with a customer, the items the vending machines are to sell and their price. Your program must accept `"Y"` or `"y"` as a **yes** and `"N"` or `"n"` as a **no**. If a customer enters anything else, your program must print `"Hum... I did not get this so I'll assume you said No."` and continue as if the customer has said **no**.

Sample runs:

```
Welcome to our Vending Machine:
Wrap -> $6
Muffin -> $3
Juice -> $2
Apple -> $1
Would you like ...
a Wrap (y/n)? N
a Muffin (y/n)? Y
a Juice (y/n)? y
a Apple (y/n)? n
Your total is $5.
```

```
Welcome to our Vending Machine:
Wrap -> $6
Muffin -> $3
Juice -> $2
Apple -> $1
Would you like ...
a Wrap (y/n)? Y
a Muffin (y/n)? y
a Juice (y/n)? Y
a Apple (y/n)? y
Your total is $12.
```

```
Welcome to our Vending Machine:
Wrap -> $6
Muffin -> $3
Juice -> $2
Apple -> $1
Would you like ...
a Wrap (y/n)? Y
a Muffin (y/n)? n
a Juice (y/n)? yap
Hum... I did not get this so I'll
assume you said No.
a Apple (y/n)? n
Your total is $6.
```

BONUS (1 mark): Add 15% tax to the total:

```
Welcome to our Vending Machine:
Wrap -> $6
Muffin -> $3
Juice -> $2
Apple -> $1
Would you like ...
a Wrap (y/n)? y
a Muffin (y/n)? y
a Juice (y/n)? n
a Apple (y/n)? y
Your total is $10.
With tax, your total is $11.5.
```

Note that you are not told which Python statements to use in order to implement this program. You need to use your own judgment and your software development experience acquired so far in this course to make the proper decisions.

Write your **VendingMachine** program below and on the next page (if needed).

Possible Solution:

```
# VendingMachine.py
#
# Midterm D300 - Part 2 - Question 2
#
# Description: Vending machine program.
#              Allows customers to select various items
#              sold in the vending machine and computes
#              the total cost of the selected item(s).
#
# BONUS (1 mark): Add 15% tax to the total.
#
# Author: AL
# Date: Feb. 2024

# Set up the variables:
# "menu", "cost", and running sum variable "total"
menu = ["Wrap", "Muffin", "Juice", "Apple"]
cost = [6, 3, 2, 1]
total = 0

# Print the menu and associated item cost
print("Welcome to our Vending Machine:")
for i in range(len(menu)):
    print(f"{menu[i]} -> ${cost[i]}")

# Ask the user ...
print("Would you like ... ")
```

Part 2 - Question 2 (cont'd)

```
# ... if s/he would like each item on the menu
for item in range(len(menu)) :
  response = input(f"a {menu[item]} (y/n)? ").lower()
    # If the user says yes ...
  if response == 'y': # in ["Y", "y"]:
    # Add the cost of the item to the running sum "total"
    total += cost[item]
  # If the user has neither said yes nor no ...
  elif response != 'n': # not in ["N", "n"]:
    # ... state so ...
    print("Hum... I did not get this so I'll assume you said No.")

# Print the total cost of selected item(s)
print(f"Your total is ${total}.")

# BONUS PART:
# Compute the tax
total *= 1.15

# Print the total cost of selected item(s) with tax
print(f"With tax, your total is ${total}.")
```