Simon Fraser University
Computing Science 120
Midterm Examination – 2
Part 1 – Theory and Understanding
Summer 2019

SOLUTION

Time: 45 minutes

Last/Family Name (please, print):

First Name (please, print):

Student Number:

Signature:

Instructor: Anne Lavergne

This examination has 8 pages.

Make sure you have a complete paper.

Instructions:

✓ No books, calculators, computers, cell phones, or other materials may be used.

✓ Read each question carefully before answering it.

✓ This exam uses Python 3.

✓ The marks for each question are given in [ ]. Let us use this to manage our time:
  • 1 mark corresponds to 1 minute of work.

22 questions
2 marks each
Total: 44 marks

The total grade for this Midterm 2 – Part 1 will be posted on CourSys.

Good luck!
Please, answer on the **Scantron sheet** – using a pencil.

1. How many times will the *statement_s* be executed in the following Python code fragment?

   ```python
   for index in range(n):
       statement_1
       statement_2
       ...
       statement_s
   ```

   A. An infinite number of times since this is an infinite loop.
   B. n times
   C. n-1 times
   D. n+1 times
   E. None of the above

2. What does the following Python code fragment produce?

   ```python
   var1 = False
   var2 = False
   var3 = False
   var4 = "Done!"
   if var1:
       print('var1')
   elif var2:
       print('var2')
   else:
       if var3:
           print('var3')
   print(var4)
   ```

   A. var3
   B. var2
   C. var4
   D. An error
   E. None of the above **-> Python code fragment prints Done! on the screen.**

3. What is the result of the following Python expression: \( \frac{21}{3} \)?

   A. There is an error in the Python expression.
   B. 7.0
   C. 7
   D. '7.0'
   E. None of the above
4. What does *hand tracing our code* mean?

A. It means that we press on the green button labelled *Run* on Repl.It in order to execute our program.

B. It means we go through the statements of our program “executing” it as if we were a computer.

C. It means we type each statement of our program on the Python Interpreter Shell and observe what it does.

D. All of the above

E. None of the above

5. Consider the list `oddNumbers = [0, 1, 3, 5, 7, 9, 11]`, which Python statement(s) would produce `[3]` as a result?

A. `oddNumbers[3]`

B. `oddNumbers[2]`

C. `oddNumbers[2:3]`

D. All of the above

E. None of the above

6. Consider the following Python code fragment:

```python
numOfPens = 2
numOfPens = numOfPens + 1
numOfPens = numOfPens - 2
print( numOfPens )
```

What does it produce as a result?

A. 1

B. 2

C. 5

D. An error -> `numOfPens` is undefined since it is not the same as `numOfPens`

E. None of the above

7. Is the following Python code fragment `return x <= y <= 67` syntactically correct?

A. Yes, it is syntactically correct and it returns the value `True` if `x = 23` and `y = 56`.

B. Yes, it is syntactically correct and it returns the value `False` if `y = 56` and `x = 23`.

C. Yes, it is syntactically correct but it returns `'None'`.

D. No. It contains a syntax error.

E. None of the above
8. Why do we create functions when we develop software?
   A. Because functions make our program easier to implement, test, debug and read.
   B. Because functions allow code to be re-used in programs.
   C. Because functions allow us to eliminate repeated code fragments.
   D. All of the above
   E. None of the above

9. Is the following Python code fragment `return 'A' <= y <= 'Z'` syntactically correct?
   A. Yes, it is syntactically correct, but returns 'None' since we cannot compare letters.
   B. Yes, it is syntactically correct and the value False is returned if y = 'B'.
   C. Yes, it is syntactically correct and it uses the ASCII values of the letters to evaluate the comparison.
   D. No. It contains a syntax error.
   E. None of the above

10. When an argument is passed to a function (during a function call), that argument must have the same name as its matching parameter?
    A. Yes
    B. It can, but it does not have to.
    C. No
    D. All of the above
    E. None of the above

11. What is not included in a function header?
    A. Function name
    B. Parameters
    C. Function body
    D. Keyword `def`
    E. None of the above

12. A function (let’s called it `func( )`) can be called from …
    A. The main part of the program if the function `func( )` is defined above the main part of the program.
    B. Another function as long as all functions (including `func( )`) are defined above the main part of the program.
    C. The main part of the program if the function `func( )` is defined below the main part of the program.
    D. All of the above
    E. A. and B.
13. How many times will the recursive function `factorial` below be called if we call it with the test data 7 from the main part of the program?

```python
def factorial(n):
    if n == 1 or n == 0:
        result = 1
    else:
        result = n * factorial(n-1)
    return result
```

A. An infinite number of times
B. 7 times
C. n-1 times
D. n+1 times
E. None of the above

14. Imagine the recursive function `factorial` (displayed in Question 13.) is called from the main part of the program with the test data 5. What is the value returned from the 4th execution of `factorial` to its 3rd execution?

A. 120
B. 24
C. 6
D. 2
E. None of the above

15. Imagine the recursive function `factorial` (displayed in Question 13.) is called from the main part of the program with the test data 8. What is the value of the variable `result` during the 8th execution of `factorial`?

A. 40320 (i.e., 8!)
B. 8
C. 0
D. The recursive function `factorial` is not called/executed 8 times.
E. None of the above

16. Which of the following is NOT true?

A. `while` loops always increment the iterating variable automatically in each iteration.
B. A `while` loop is best for situations when we do not know how many times the statements in the body of the loop need to be executed.
C. The statements in the body of a `for` loop must be indented.
D. Variables declared in a function have local scope.
E. None of the above
17. What does this Python code fragment output?

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

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

# Main part of program
aNumber = 5
result = square(aNumber)
print(result + 23)
```

A. 33
B. 48
C. 25
D. It doesn't run, it produces an error.
E. None of the above

18. What does this Python code fragment produce?

```python
import random

def randomColour(colours) :
    default = "white"
    color = random.choice(colours)
    return color

# Main part of program
print(default)
print(randomColour(["blue", "purple", "green"]))
```

A. white
   blue
B. white
   purple
C. white
   green
D. The code does not execute, it produces an error. -> default not defined
E. None of the above
19. In the Python code fragment below, fill in the blank with the most appropriate data (from A., B., C., or D.) such that the Python code fragment prints the word *Secret*.

```python
def encode(message, rules):
    secretMessage = ""
    for i in range(len(message)):
        secretMessage += rules[message[i]]
    return secretMessage

# Main part of program
# Choose the encodingRules that will create "Secret"
encodingRules = ____________________________
print(encode("Hello!", encodingRules))
```

A. ("S", "e", "c", "r", "e", "t")
B. {0:"S", 1:"e", 2:"c", 3:"r", 4:"e", 5:"t"})
C. ["H", "e", "L", "l", "o", "]
D. {"H":"S", "e":"e", "L":"c", "l":"r", "o":"e", "]
E. None of the above

20. Which turtle drawing would the Python code fragment below produce?

```python
import turtle
def drawSpiral(myTurtle, lineLen):
    if lineLen > 0:
        myTurtle.forward(lineLen)
        myTurtle.right(90)
        drawSpiral(myTurtle, lineLen-5)

# Main part of program
myTurtle = turtle.Turtle()
drawSpiral(myTurtle,100)
```

A. [Image of a square spiral]
B. [Image of a spiral with arrows]
C. [Image of a spiral with a dot in the center]
D. [Image of a spiral with a dot in the center]
E. None of these drawings
21. What does this Python code fragment output?

```python
def funnyFruits(aList):
    if len(aList) < 1:
        result = False
    elif len(aList) < 4:
        result = True
    else:
        result = False
    return result

# Main part of program
fruits = ["Peach"]
while funnyFruits(fruits):
    print("Apple")
    fruits.append("Orange")
print("Cranberry")
```

<table>
<thead>
<tr>
<th>A.</th>
<th>B.</th>
<th>C.</th>
<th>D.</th>
<th>E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peach</td>
<td>Peach</td>
<td>Apple</td>
<td>Apple</td>
<td>None of the above</td>
</tr>
<tr>
<td>Apple</td>
<td>Apple</td>
<td>Apple</td>
<td>Apple</td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>Apple</td>
<td>Apple</td>
<td>Orange</td>
<td></td>
</tr>
<tr>
<td>Orange</td>
<td>Cranberry</td>
<td>Cranberry</td>
<td>Orange</td>
<td></td>
</tr>
<tr>
<td>Cranberry</td>
<td></td>
<td></td>
<td>Cranberry</td>
<td></td>
</tr>
</tbody>
</table>

22. How many times does the inner `for` loop in the Python code fragment below execute?

```python
def something(listNum):
    for num1 in listNum:
        for num2 in listNum:
            if num1 > num2:
                return num1 * num2

# Main part of program
print(something([5,8,9,2,5]))
```

<table>
<thead>
<tr>
<th>A.</th>
<th>B.</th>
<th>C.</th>
<th>D.</th>
<th>E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>None of the above</td>
</tr>
</tbody>
</table>