Quiz 01 - Practice

COMP 110: Introduction to Programming
Spring 2024

Thursday 15, 2024

Name: ______________________________

9-digit PID: ______________________________

Do not begin until given permission.

Honor Code: I have neither given nor received any unauthorized aid on this quiz.

Signed: ______________________________

Question 1: Multiple Choice Completely fill in the bubble next to your answer using a pencil. Each question should have exactly one filled-in bubble.

1.1. The following string is an example of an format string:

```
"{1 + 1}"
```

- ◯ `True`
- ◯ `False`

1.2. What does the following string evaluate to?

```
"\\\\"
```

- ◯ `\`
- ◯ `\\`
- ◯ `\\\`
- ◯ `\\\\`

1.3. What is the printed output of the following `print` function call?

```
print(f"C{'OM'}P{100 + 10}")
```

- ◯ `fCOMP10010`
- ◯ `COMP110`
- ◯ `C'OM'P100 + 10`
- ◯ `Error: Invalid Syntax`

1.4. What does the `chr` function do in the following example:

```
chr(129312)
```

- ◯ `Converts an int representation into a string character`
- ◯ `Converts a string character into an int representation`
- ◯ `Chars a number by burning it just a little`
- ◯ `Error: This function is not built-in to Python`

1.5. What is the *type* and *evaluation* of this expression in Python?

```
"ABCD" < "ABCZ"
```

- ◯ `False`
- ◯ `True`

1.6. Hexidecimal is base-16, binary is base-2, and decimal is base-10.

- ◯ `False`
- ◯ `True`

1.7. Which operator has the highest precedence in an expression?

- ◯ or
- ◯ $>$
- ◯ $+$
- ◯ and
- ◯ not

1.8. What is the evaluation of the following expression:

```
1 > 0 or "B" > "A"
```

- ◯ False
- ◯ True

1.9. What is the evaluation of the following expression:

```
"A" == "B" and "B" == "C"
```

- ◯ False
- ◯ True

1.10. A Tuple can hold 0, 1, or more values:

- ◯ False
- ◯ True

1.11. What is the evaluation of the following expression:

```
(110,210,301)[1 + 1]
```

- ◯ 0
- ◯ 1
- ◯ 110
- ◯ 210
- ◯ 301
- ◯ Error

1.12. What is the evaluation of the following Python expression?

```
not True or True
```

- ◯ `False`
- ◯ `True`

1.13. What is the evaluation of the following expression?

```
(1,) + (1, 0)
```

- ◯ `(1, 0)`
- ◯ `(2, 0)`
- ◯ `(1, 1, 0)`
- ◯ `Error`

1.14. Which of the following are required in a recursive function that does not infinitely recur?

- ◯ A base case without a recursive function call
- ◯ Arguments changing in recursive case
- ◯ Recursive cases make progress toward the base case
- ◯ All of the above

1.15. Which of the following is a valid function call to the following function signature?

```
def a_func(x: int, y: int = 0)
    -> int:
  ...
```

- ◯ A. `a_func()`
- ◯ B. `a_func(1)`
- ◯ C. `a_func(1, 2)`
- ◯ B and C
- ◯ None of the above

1.16. What type of error occurs when recursion appears to infinitely

- ◯ `Name Error`
- ◯ `Index Error`
- ◯ `Stack Overflow Error`
- ◯ `Syntax Error`

1.17. What will the following Python expression evaluate to?

```
1 + True
```

- ◯ `True`
- ◯ `2`
- ◯ `1`
- ◯ `False`

1.18. What is the following statement declaring?

```
PI: float = 3.14
```

- ◯ Global Named Constant
- ◯ Local Named Constant
- ◯ Either of the above, depending on where it is declared
- ◯ None of the above

1.19. Consider the following function declaration:

```
def a_func(x: int, y: int = 0)
    -> int:
  ...
```

Which of the following are valid ways of calling the function?

- ◯ A. `a_func(x=1, y=2)`
- ◯ B. `a_func(x=1)`
- ◯ C. `a_func(1, 2)`
- ◯ A and B
- ◯ A, B, and C
- ◯ None of the above

1.20. What does the built-in `id` function evaluate to when called?

- ◯ The part of a computer's brain an object is in.
- ◯ The ID, which is the memory address, of the argument it is given.
- ◯ The *identity* of its argument, e.g. the argument itself

Question 2: Respond to the following questions

Consider the following code listing:

```
def eight_ball(choice: int) -> str:
  """Returns an 8-ball response."""
  if choice <= 0:
    return "Unlikely."
  else:
    if choice > 0:
      return "It is certain."
    else:
      return "Ask again later."
```

2.1. Write a function call expression to the `eight_ball` function that evaluates to `"It is certain."`

2.2. Write a function call expression to the `eight_ball` function that evaluates to `"Unlikely."`

2.3. Write a function call expression to the `eight_ball` function that evaluates to `"Ask again later."`

2.4. What value and type does the following expression evaluate to: `3 + 4 == 6`

2.5. What value and type does the following expression evaluate to?

```
((True and False) or (False or True)) != False
```

2.6. What value and type does the following expression evaluate to? (This is a notably obtuse expression, but breaking it down and simplifying it will help you reinforce your understanding of expressions with subscription notation.)

```
(1, 2, 3)[(0, 1, 2)[1 - int("012"[1])]]
```

Question 3: Memory Diagram Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```
def gen(stop: int, acc: tuple[int, ...] = (), i: int = 0) -> tuple[int, ...]:
  """Generate a tuple from i to stop."""
  if i >= stop - 1:
    return acc + (i,)
  else:
    return gen(stop, acc + (i,), i + 1)


print(gen(3))
```

Output

Stack

Globals

Heap

Question 4: Memory Diagram Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```
LETTERS: tuple[str, ...] = ("A", "E", "F", "H", "K", "L", "Z")


def search(needle:str, haystack:tuple[str, ...], min: int, max: int) -> int:
  """Find the index of a needle in a sorted haystack, or -1 if not found."""
  if min > max:
    return -1
  else:
    MIDDLE: int = ((max - min) // 2) + min
    print(f"Guess: {MIDDLE}")
    if needle == haystack[MIDDLE]:
      return MIDDLE
    elif needle > haystack[MIDDLE]:
      return search(needle, haystack, MIDDLE + 1, max)
    else:
      return search(needle, haystack, min, MIDDLE - 1)


print(search(needle="K", haystack=LETTERS, min=0, max=len(LETTERS) - 1))
```

Output (You can write successive lines beside one another separated by a //)

Stack

Heap

Globals

4.1. Knowing that the `haystack` `tuple` is *sorted* in ascending order, describe the general strategy this algorithm takes for finding the index of the `needle` parameter in the `haystack`.

4.2. On the previous code listing, what lines do you find the `return` statements of the *base cases* of the `search` function?

4.3. On the previous code listing, what lines do you find the `return` statements of the *recursive cases* of the `search` function?

4.4. One of the conditions for writing a recursive function that is not infinite is that the recursive cases make progress toward the base case(s). How do the recursive cases make progress toward the base case resulting in `-1`?

Question 5: Memory Diagram Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```python
def fib(n: int) -> int:
  """Compute the fibonacci of n"""
  print(f"fib({n})")
  if n == 0 or n == 1:
    return n
  else:
    N1: int = fib(n - 1)
    N2: int = fib(n - 2)
    return N1 + N2

print(fib(3))
```

Output

Stack

Globals

Heap

This page intentionally left blank. Do not remove from quiz packet.