CS 320 - Spring 2023

Instructor: Meenakshi Syamkumar

Exam 1 — 13%

(Last) Surname: _____ (First) Given name: _____

NetID (email): _____ @wisc.edu

Fill in these fields (left to right) on the scantron form (use #2 pencil):

1. LAST NAME (surname) and FIRST NAME (given name), fill in bubbles
2. IDENTIFICATION NUMBER is your Campus ID number, fill in bubbles
3. Under *ABC* of SPECIAL CODES, write your lecture number, fill in bubbles:
   001 - MWF 11:00am
   002 - MWF 1:20pm
4. Under **F** of SPECIAL CODES, write **6** and fill in bubble **6**

If you miss step 4 above (or do it wrong), the system may not grade you against the correct answer key, and your grade will be no better than if you were to randomly guess on each question. So don't forget and double check it's correct!

You may only reference your note sheet. You may not use books, calculators, or other electronic devices during this exam. You may not sit near your friends or look at your neighbors during this exam. Please place your student ID face up on your desk. Turn off and put away portable electronics (including smart watches) now.

Use a #2 pencil to mark all answers. DO NOT USE PEN on the scantron.

When you're done, please hand in the exam and note sheet and your filled-in scantron form. The note sheet will not be returned.

(Blank Page)

1. What is the output of the below code snippet?

```
def mystery(some_nums):
    if len(some_nums) == 0:
        return []
    else:
        return [some_nums.pop(-1)] + mystery(some_nums)

some_nums = [5, 2, 7, -1]
print(mystery(some_nums))
```

A. `[-1, 7, 2, 5]`

B. `[5, 2, 7, -1]`

C. `[-1, 2]`

D. `[7, 5]`

E. `RecursionError`

2. Consider the below code snippet.

```
class Car:
    def __init__(self, make, models):
        self.make = make
        self.models = models

cars = Car("Toyota", ["Avalon", "Corolla", "Sienna"])
print(len(cars)) # line 7
```

Which of the following special methods must be implemented for # line 7 to produce 3 as the output?

A. `len` B. `_repr_svg_` C. `__getitem__` **D.** `__len__` E. `for`

3. Consider the below code snippet. How many attributes will the object instance referenced by `cars` have?

```
class Car:
    def __init__(self, make, models, colors):
        self.make = make
        self.models = models
        year = 2023
        ranking = 3
        color = colors


cars = Car("Toyota", ["Avalon", "Corolla", "Sienna"], \
                        ["red", "green", "blue", "gray"])
```

A. 2 B. 3 C. 4 D. 5

4. What numbers get printed by the following code snippet?

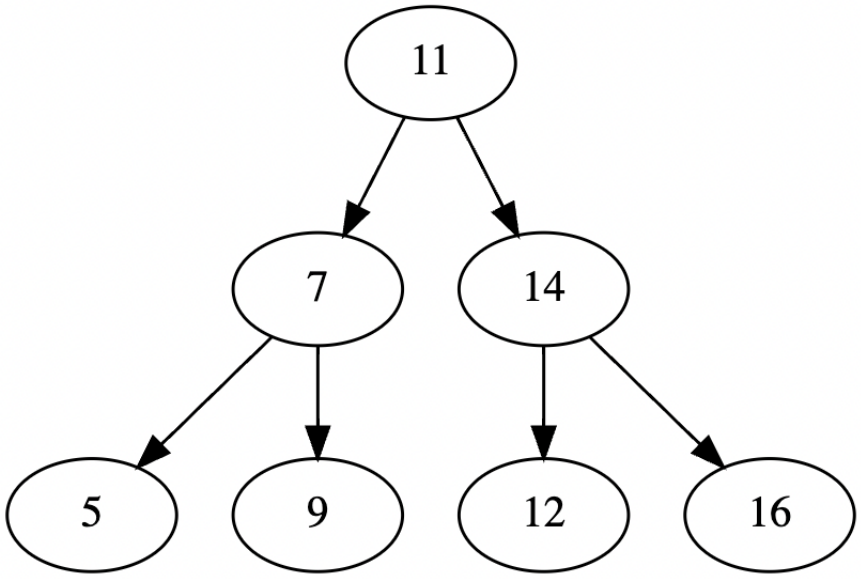
```
def mystery():
    a = 0
    b = 1

    while True:
        yield a
        temp = a + b
        a = b
        b = temp

f = mystery()
print(next(f))
print(next(f))
print(next(f))
```

A. 0, 1, 1 B. 0, 1, 2 C. 1, 1, 2 D. 1, 2, 3

5. Consider the BST insertion algorithm we learned in class. Given the below BST, which of the following **CANNOT** be the insertion order? For every node, consider first child as left and second child as right.



A. `[11, 5, 7, 14, 9, 12, 16]`

B. `[11, 7, 14, 5, 9, 12, 16]`

C. `[11, 7, 5, 9, 14, 12, 16]`

D. `[11, 14, 7, 12, 9, 5, 16]`

6. Consider the below code snippet.

```
class TrafficLight:
    def __init__(self, color, distance):
        self.color = color
        self.distance = distance

tl1 = TrafficLight("green", 10) # line 6
```

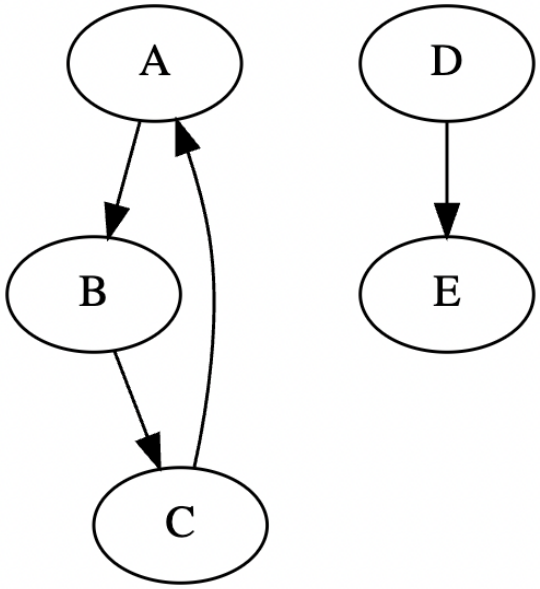
How many arguments are passed on # line 6?

A. 0 B. 1 C. 2 **D. 3**

7. Which of the following will enable us to **efficiently** implement a queue for BFS?

A. `set` B. `list` **C.** `deque` D. `heapq` E. `stack`

8. What can be said about the following graph?



A. cyclic but not connected

B. cyclic and connected

C. acyclic but not connected

D. acylic and connected

9. Which complexity class is worst / slowest among the following choices?

A. `O(log N)` B. `O(N)` **C.** `O(N**2)` D. `O(N log N)`

10. Which of the following implicitly invokes `__le__` special method?

A. `obj1 != obj2` B. `obj1 == obj2` C. `obj1 < obj2` **D.** `obj1 <= obj2`

11. Suppose `BSTNode` class stores information about BST nodes, is the below implementation of `__getitem__` method recursive?

```
class BSTNode:
    def __init__(self, name, val):
        self.key = name
        self.val = val
        self.left = None
        self.right = None

    def __getitem__(self, target):
        if target < self.key and self.left != None:
            return self.left[target]
        elif target > self.key and self.right != None:
            return self.right[target]
        assert self.key == target
        return self.val
```

A. `True` B. `False`

12. Consider the below code snippet.

```
class Polygon:
    def __init__(self, sides):
        self.sides = sides

class Rectangle(Polygon):
    def __init__(self):
        pass # line 7

r1 = Rectangle()
```

Which of the following lines of code can be used to invoke the `Polygon` class constructor to replace `pass` on # line 7?

A. `super.__init__(4)`

B. `super().__init__(4)`

C. `self.__init__(4)`

D. `self().__init__(4)`

13. If a BST is constructed using the algorithm we learned in class, and the insert order is `[8, 3, 1, 6]`, where will `6` be?

    A. `root.left.left`

    B. `root.left.right`

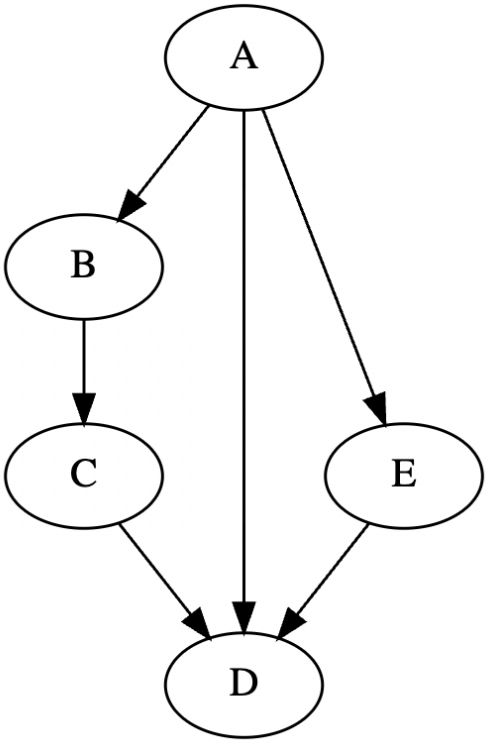
    C. `root.right.left`

    D. `root.right.right`

14. Which one of the following list operations have worst case complexity? Assume that `L` is storing a reference to a list object instance.

    A. `L.pop(-1)` **B.** `L.pop(0)` C. `L.append(1)` D. `L[len(L) // 2]`

15. Given the below graph, which of the following paths will **DFS** return between nodes `A` and `D`? Assume that for every node its children nodes are alphabetically ordered.



    A. `None` B. `(A, D)` C. `(A, E, D)` **D.** `(A, B, C, D)`

16. Considering the same graph as the previous question, which of the following paths will **BFS** return between nodes `A` and `D`? Again, assume that for every node its children nodes are alphabetically ordered.

    A. `None` **B.** `(A, D)` C. `(A, E, D)` D. `(A, B, C, D)`

17. What is printed?

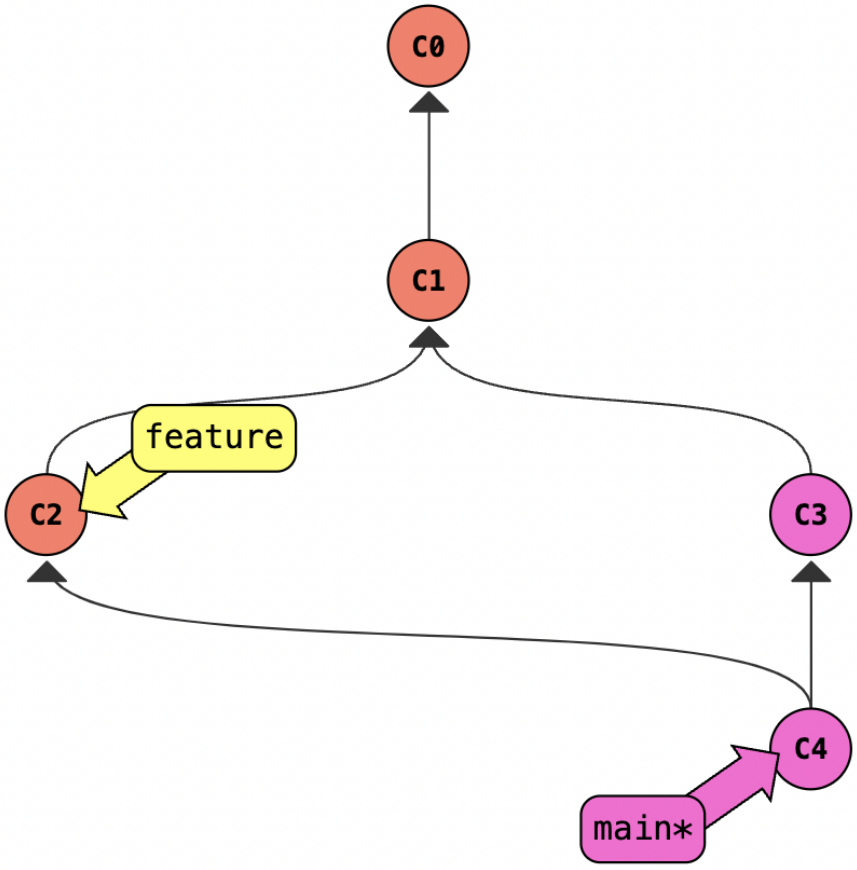
```
import heapq

items = []
for val in [10, 3, 1, 5, 21]:
    heapq.heappush(items, val)

print(heapq.heappop(items))
```

A. 1 B. 3 C. 5 D. 10 E. 21

18. Given the below git commit graph, which of the following git commands was executed last?



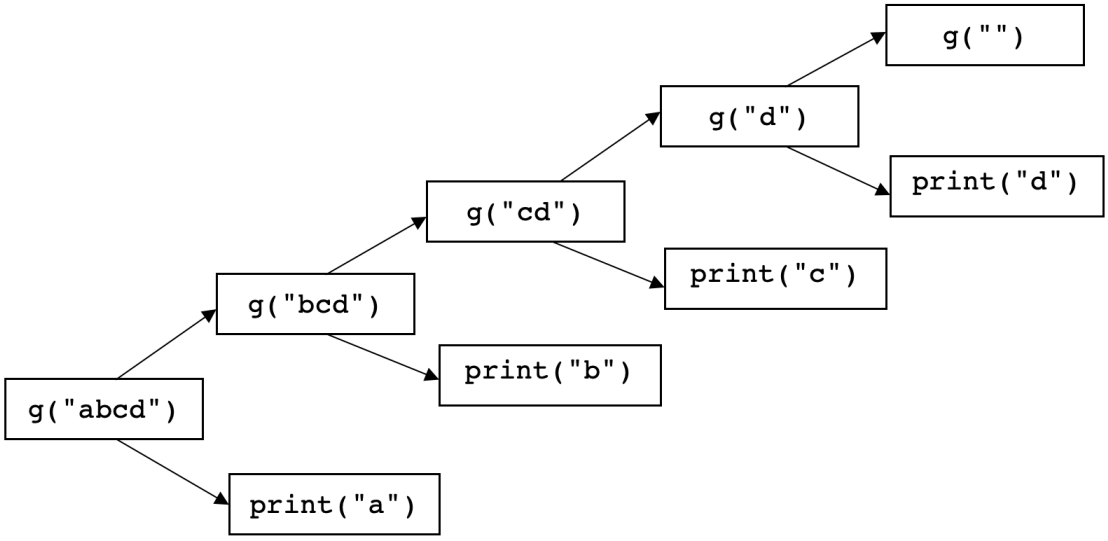
A. `git tag`

B. `git merge feature`

C. `git commit`

D. `git merge main`

19. Consider the below call graph. What gets printed **first**?



A. `a` B. `b` C. `c` **D.** `d`

20. Which of the following is the correct invocation of `check_output` for executing `git checkout` command inside a directory called `some_repo`? Assume that branch `f1` exists.

A. `check_output("git checkout f1", cwd="some_repo")`

B. `check_output("git checkout f1", pwd="some_repo")`

C. `check_output(["git", "checkout", "f1"], cwd="some_repo")`

D. `check_output(["git", "checkout", "f1"], pwd="some_repo")`

(Blank Page)