## CS 220 - Spring 2022
### Instructors: Meenakshi Syamkumar, Andrew Kuemmel, Cole Nelson

### Exam 2 — 10%

(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 (Meena)
    - 002 - MWF 1:20pm (Meena)
    - 003 - MWF 8:50am (Andy)
    - 004 - MWF 9:55am (Cole)
4. Under *F* of SPECIAL CODES, write **A** 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!**

---

Many of the problems in this exam are related to the course projects, but some questions assume the availability of slightly different functions (e.g., for accessing the data). We won't have any trick questions where we call a function that doesn't exist and you need to notice. Thus, if you see a call to a function we haven't explicitly defined in the problem, assume the function was properly implemented (perhaps immediately before the code snippet we DO show) and is available to you.

You may only reference your note sheet. You may not use books, your neighbors, calculators, or other electronic devices on 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. When you're done, please hand in these sheets in addition to your filled-in scantron.

(Blank Page)

# General

1. Which of the following is immutable?

   A. list    **B. string**    C. dictionary    D. none of the above

2. What is the output of the following code snippet?

```
s = "Hello World!"
s = s.upper().split(" ")
print(s[-1][-1])
```

   A. WORLD!    B. D    **C. !**    D. IndexError

3. What is the output of the following code snippet?

```
my_dict = {1: "One", 2: "Two", 3: "Four"}
my_dict["Zero"] = 0
print(1 in my_dict and 0 in my_dict)
```

   A. True    **B. False**    C. None    D. KeyError

4. What is the output of the following code snippet?

```
my_list = [1, 2, 3]
my_list.append("one")
my_list.extend([11, 22, 33, 44, 55])
print(my_list.index("one") + len(my_list))
```

   A. 8    B. 9    C. 11    **D. 12**

5. What is the type of the return value of the below `process_csv` function?

```
import csv
def process_csv(filename):
    example_file = open(filename, encoding="utf-8")
    example_reader = csv.reader(example_file)
    example_data = list(example_reader)
    example_file.close()
    return example_data
```

   A. tuple    B. string    **C. list of lists**    D. list of dictionaries

6. What gets printed after executing this code snippet?

```python
def some_func(some_list):
    some_list[1] = ["peach"]

list_a = ["apple", "orange", "banana"]
some_func(list_a)
list_a[0] = "pear"
print(list_a)
```

    A. ["apple", "orange", "banana"]

    B. ["pear", "orange", "banana"]

    **C.** ["pear", ["peach"], "banana"]

    D. ["pear", "peach", "banana"]

7. What gets printed after executing this code snippet?

```python
import copy
list_a = ["apple", ["orange", "banana"]]
list_b = copy.copy(list_a)
list_b[0] = "pear"
list_b[1][1] = "peach"
print(list_a)
```

    A. ["apple", ["orange", "banana"]]

    **B.** ["apple", ["orange", "peach"]]

    C. ["pear", ["orange", "banana"]]

    D. ["pear", ["orange", "peach"]]

8. What gets printed after executing this code snippet?

```python
import copy
list_a = ["apple", ["orange", "banana"]]
list_b = copy.deepcopy(list_a)
list_b[0] = "pear"
list_b[1][1] = "peach"
print(list_a)
```

    **A.** ["apple", ["orange", "banana"]]

    B. ["apple", ["orange", "peach"]]

    C. ["pear", ["orange", "banana"]]

    D. ["pear", ["orange", "peach"]]

9. What is the output of this code snippet?

```
def some_func(some_list):
    some_list = some_list + ["strawberry", "blackberry", "raspberry"]

list_a = ["apple", "orange", "banana"]
some_func(list_a)
print(list_a[3][-5:])
```

A. "anana"   B. "berry"   C. "strawberry"   **D.** IndexError

10. What is the output of the following code snippet?

```
def my_func(n):
    if n > 5:
        return n + my_func(n - 2)
    else:
        return -1
print(my_func(10))
```

A. -1   B. 5   C. 13   **D. 23**   E. RecursionError

11. What is the output of the following code snippet?

```
def get_diff(city_tuple):
    return len(city_tuple[1]) - len(city_tuple[0])

names = [("Madison", "Chicago"),
         ("Chicago", "Los Angeles"),
         ("Los Angeles", "Seattle")]

names.sort(key = get_diff)
print(names[0])
```

   A. ("Madison", "Chicago")

   B. ("Chicago", "Los Angeles")

   **C.** ("Los Angeles", "Seattle")

   D. None of the above due to an IndexError

12. Which of the following will produce a sorted dictionary based on length of the city names?

```
cities = {"New York": 1, "Chicago": 2, "Los Angeles": 3}
```

    **A.** `dict(sorted(cities.items(), key = lambda c:  len(c[0])))`

    B. `dict(sorted(cities.items(), key = lambda c:  len(c[1])))`

    C. `sorted(cities, key = len)`

    D. `cities.sort(key = lambda c:  len(c[0]))`

13. Which of the following will return a list of names with a length longer than 7?

```
city_names = ["New York", "Chicago", "Los Angeles"]
```

    A. `[len(n) for n in city_names if len(n) > 7]`

    B. `[len(city_names) for n in city_names if len(city_names) > 7]`

    **C.** `[n for n in city_names if len(n) > 7]`

    D. `[city_names for n in city_names if len(city_names) > 7]`

14. What is the output of the following code snippet?

```
original = ["a", "b"]
new = []
for i in range(len(original) + 1):
    try:
        new.append(original[i])
    except:
        new.append(-1)
print(new)
```

    A. `["a", -1]`

    B. `["a", "b"]`

    C. `[-1, "a", "b"]`

    **D.** `["a", "b", -1]`

15. What is the type of `data` and `split_data` in the following code?

```
f = open("madison.txt") # assume that this file contains some lines
data = f.read()
split_data = data.split("\n")
f.close()
```

        **A. data is a string and split_data is a list**

        B. data is a string and split_data is a string

        C. data is a file and split_data is a list

        D. data is a file and split_data is a string

## Covid-19 vaccinations

For each question in this section, assume that the initial code executed is as follows:

```
vaccinations = [
    {"iso": "ALB", "vacc": "Pfizer, Sinovac",
        "vacc months": ["Jan", "Feb", "Mar", "May"],
        "vacc counts": [549, 6728, 110015, 650001]},
    {"iso": "FRO", "vacc": "Moderna, Pfizer",
        "vacc months": ["Apr"], "vacc counts": [15531]},
    {"iso": "VEN", "vacc": "Sputnik V",
        "vacc months": ["Feb", "Apr", "May", "Jun"],
        "vacc counts": [157, 10770, 316015, 620437]},
    {"iso": "ZMB", "vacc": "Oxford",
        "vacc months": ["Jun"], "vacc counts": [148304]}
]
```

16. What does the following evaluate to? `vaccinations[0]["vacc"][1:3]`

    A. 6728, 110015   B. 6728, 110015, 650001   **C. fi**   D. fiz

17. What does the following evaluate to?

    `vaccinations[-2]["vacc months"].index("May")`

    A. 3   B. 5   C. May   **D. 2**   E. IndexError

18. What does the following evaluate to?

    `vaccinations[0]["vacc months"] + vaccinations[-1]["vacc months"]`

    A. 'Jan, Feb, Mar, May, Jun'

    **B. ['Jan', 'Feb', 'Mar', 'May', 'Jun']**

    C. ['Jan', 'Feb', 'Mar', 'May', ['Jun']]

    D. ['Mar', 'Apr', 'Jun', 'Jun']

19. What does the following evaluate to?

```
[result for result in vaccinations[0]]
```

    **A.** ['iso', 'vacc', 'vacc months', 'vacc counts']

    B. ['AFG', 'ALB', 'FRO', 'VEN', 'ZMB']

    C. ['iso']

    D. ['AFG', 'Oxford', ['Mar', 'Apr', 'Jun'], [8207, 120495, 190261]]

20. Which of the following would generate a dict mapping country `iso` to **sum** of `vacc counts` for each country?

    A. `{country['iso']:total(country['vacc counts']) for country in vaccinations}`

    B. `{country['iso']:(country['vacc counts']) for country in vaccinations}`

    C. `dict([sum(country['vacc counts']) for country in vaccinations])`

    **D.** `{country['iso']:sum(country['vacc counts']) for country in vaccinations}`

21. What will be printed when the following code is run?

```
buckets = {}
for country in vaccinations:
    for month in country["vacc months"]:
        if month not in buckets:
            buckets[month] = []
        buckets[month].append(country)
print(len(buckets), len(buckets["Jun"]))
```

    A. 0 0   B. 2 2   C. 2 4   **D. 6 2**   E. 6 4

Aassume that the initial code executed is as follows:

```
vaccinations = [
    {"iso": "ALB", "vacc": "Pfizer, Sinovac",
        "vacc months": ["Jan", "Feb", "Mar", "May"],
        "vacc counts": [549, 6728, 110015, 650001]},
    {"iso": "FRO", "vacc": "Moderna, Pfizer",
        "vacc months": ["Apr"], "vacc counts": [15531]},
    {"iso": "VEN", "vacc": "Sputnik V",
        "vacc months": ["Feb", "Apr", "May", "Jun"],
        "vacc counts": [157, 10770, 316015, 620437]},
    {"iso": "ZMB", "vacc": "Oxford",
        "vacc months": ["Jun"], "vacc counts": [148304]}
]
```

22. What will be printed when the following code is run?

```
result = 0
for country in vaccinations:
    if 'Jun' in country['vacc months']:
        result += country['vacc counts'][country['vacc months'].index('Jun')]
print(result)
```

     A. Total vaccination counts from Jan to Jun.

     B. Sum of Jun indices.

     **C. Total vaccination counts in Jun.**

     D. Number of countries that offered vaccines in Jun.

23. Which of the following will correctly complete the below code to sort `vaccinations` based on increasing order of sum of `vacc counts` for each country?

```
def extract_total(???):
    return sum(???)
sorted(vaccinations, key = extract_total)
```

     A. `vacc counts, vacc counts`

     **B. `country_dict, country_dict["vacc counts"]`**

     C. `vaccinations, vaccinations[country_dict["vacc counts"]]`

     D. `country, country[3]`

**Blank Page: this page is intentionally blank, please turn over for next section questions**

## Student Roster

For each question in this section, assume that the **initial code** executed is as follows:

```python
from collections import namedtuple
Student = namedtuple("Student", ["name", "grade", "ID"])
roster = [
    Student("Michael", "B", 2),
    Student("Sofia", "A", 3),
    Student(ID = 6, name = "Alexander", grade = "AB"),
    Student("Jill", "AB", 18)
]
```

24. What does `roster[2].ID` evaluate to?

    **A.** 6   B. "6"   C. "B"   D. 3   E. "3"

25. What is the output of the following code snippet?

```python
names = list()
for student in roster:
    names.append(student.name)
names.sort(key = lambda n: len(n))
print(names)
```

      A. ['Alexander', 'Jill', 'Michael', 'Sofia']

      **B.** ['Jill', 'Sofia', 'Michael', 'Alexander']

      C. ['Michael', 'Sofia', 'Alexander', 'Jill']

      D. ['Alexander', 'Michael', 'Sofia', 'Jill']

26. Which of the following will result in an error?

      A. `roster.append(Student("Sarah", "A", 1))`

      B. `roster[0] = Student("Alex", "AB", 4)`

      **C.** `roster[0].ID = 20`

      D. `roster.extend([Student("Cindy", "AB", 5), Student("Jack", "A", 7)])`

      E. None of the above

27. Which of the following sorts `roster` by decreasing order of IDs?

        A. `sorted(roster, key = lambda s:s.ID)`

        B. `sorted(roster, lambda s:s["ID"])`

        C. `sorted(roster, lambda s:s["ID"], reverse = True)`

        **D. `sorted(roster, key = lambda s:s.ID, reverse = True)`**

28. After executing the below code snippet, what does `roster[1].name` evaluate to?

```
import copy
new_roster1 = copy.copy(roster)
new_roster2 = roster
new_roster1[1] = Student("Rose", "B", 17)
new_roster2[1] = Student("Emily", "A", 42)
```

**A. Emily**    B. Rose    C. Sofia    D. Michael

29. Which one of the following will create a list of student names with AB grade?

        A. `{key:value for key, value in students.items() if student.grade == "AB"}`

        B. `[student["name"] for student in roster if student.grade == "AB"]`

        **C. `[student.name for student in roster if student.grade == "AB"]`**

        D. `{student.name for student in roster if student.grade == "AB"}`

## Files

30. What will be in happy.txt after this code runs?

```
f = open("happy.txt", "w")
f.write("Don't worry, ")
f.close()
f = open("happy.txt", "w")
f.write("Be ")
f.write("Happy")
f.close()
```

        A. `"Don't worry, Be Happy"`

        **B. `"Be Happy"`**

        C. `"Don't worry, \nBe \nHappy"`

        D. `"Happy"`

31. Which of the following is a valid JSON?

      A. `{"77":{"name":"Kirby", "rating":None}}`

      **B.** `{"77":{"name":"Kirby", "rating":null}}`

      C. `{'77':{'name':'Kirby', 'rating':None}}`

      D. `{'77':{'name':'Kirby', 'rating':null}}`

32. What is the default delimiter between rows in a CSV file?

    **A. newline**    B. comma    C. space    D. semicolon

## Errors

33. What call to foo will **not** cause an AssertionError?

```
def foo(word, num):
    assert type(word) == str and type(num) == int
    assert len(word) > num
    return (word + " ") * num
```

    A. `foo("monkey", 6)`

    B. `foo("koala", "3")`

    C. `foo(["badger"], "4")`

    **D.** `foo("penguin", 5)`

    E. Both A and D.

34. What is the output of the following code?

```
input_data = [4, -3, "two", 6]
output_data = []
for data in input_data:
    try:
        output_data.append(data + 2)
    except:
        output_data.append("error")
print(output_data)
```

    A. [6, -1, 'error']

    **B. [6, -1, 'error', 8]**

    C. [6, -1, 'two2', 8]

    D. ['error', 'error', 'error', 'error']

    E. TypeError

35. Which **lines of code** will get executed in the following code?

```
def b():
    print("Start: B")      # line 1
    z = 100 / 0            # line 2
    print("End: B")        # line 3

def a():
    print("Start: A")      # line 4
    try:                   # line 5
        b()                # line 6
    except:                # line 7
        print("B failed")  # line 8
    print("End: A")        # line 9

try:                       # line 10
    a()                    # line 11
except:                    # line 12
    print("A failed")      # line 13
```

    A. 1, 2, 3, 4, 5, 6, 9, 10, 11

    **B. 1, 2, 4, 5, 6, 7, 8, 9, 10, 11**

    C. 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

    D. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

(Blank Page)

(Blank Page: you may tear this off for scratch work, but hand it in with your exam)

(Blank Page: you may tear this off for scratch work, but hand it in with your exam)