# CS 220 - Fall 2023

Instructors: Mike Doescher, Gurmail Singh, and 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 08:50 AM (Mike)
   - 002 - MWF 11:00 AM (Mike)
   - 003 - MWF 01:20 PM (Gurmail)
   - 004 - MWF 03:30 PM (Gurmail)
   - 005 - MWF 08:50 AM (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!**

You may only reference your note sheet. You cannot use books, your neighbors, calculators, or other electronic devices 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. 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.

## General

1. What is the output of the following code?

```
my_dict = {0: "zero", 1: "one"}
my_dict[2] = "two"
my_dict.pop(0)
my_dict.pop(3, None)
print(list(my_dict.keys()))
```

   A. `[0, 1]`
   B. `[1]`
   C. `[0, 1, 2]`
   D. `[1, 2]`
   E. `This code will throw an error`

2. In Python dictionaries, which statement is true?

   A. `Keys can be anything, values can be anything`
   B. `Keys can be anything, values must be immutable`
   C. `Keys must be immutable, values can be anything`
   D. `Keys must be immutable, values must be immutable`

3. What is the output of the code?

```
def func(numbers):
    n = numbers.pop(-1)
    return n

num_list = [1, 2, 3, 4]
n = func(num_list)
print(n, num_list)
```

   A. `1, [1, 2, 3, 4]`
   B. `1, [2, 3, 4]`
   C. `4, [1, 2, 3, 4]`
   D. `4, [1, 2, 3]`
   E. `None, [1, 2, 3, 4]`

4. What is the output of the code?

```
def add(a, b):
    a = a + b
    return a

x = 1
y = 2
list1 = [1, 2]
list2 = [3, 4, 5]
add(x, y)
add(list1, list2)
print(x, list1)
```

   A. `3, [1, 2, 3, 4, 5]`
   B. `1, [1, 2]`
   C. `1, [1, 2, 3, 4, 5]`
   D. `3, [1, 2]`
   E. `This code will throw an error`

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

```
string1 = "CS begins with"
string2 = "Hello World!"
s = string1 + " " + string2
print(s[:2] + s[-6:-1])
```

   A. `"CSWorld"`
   B. `"CSWorld!"`
   C. `" World"`
   D. `"CS World!"`
   E. `None of the above`

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

```
string = "Great match!"
string = string.lower().split()
print(string[-1][0])
```

   A. "G"

   B. "g"

   C. "m"

   D. "!"

   E. This code will throw an error

7. Fibonacci numbers are a sequence of numbers starting with 0 and 1 where each number after that is the sum of the two preceding ones. To illustrate, here are the indices and the values of first seven Fibonacci numbers:

| **Index** | **Value** |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |
| 5 | 5 |
| 6 | 8 |

Fill in the blanks to complete a recursive function for calculating the Fibonacci number at index `n`.

```
def Fib(n):
    if [blank 1]:
        return n
    else:
        return [blank 2]
```

   A. `blank 1:  n==0`
      `blank 2:  Fib(n-1) + Fib(n-2)`

   B. `blank 1:  n==0 or n==1`
      `blank 2:  Fib(n-1) + Fib(n-2)`

   C. `blank 1:  n==0`
      `blank 2:  Fib(n-1) + Fib(n-2) + 1`

   D. `blank 1:  n==0 or n==1`
      `blank 2:  Fib(n-1) + Fib(n-2) + 1`

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

```
def func(a, b):
    if a <= b:
        return a + b
    else:
        return a + b + func(a-1, b+1)

print(func(6, 2))
```

   A. `24`

   B. `28`

   C. `18`

   D. `16`

   E. `This code will cause a RecursionError`

9. What is the default row terminator for a CSV file?

   A. `space`

   B. `newline`

   C. `comma`

   D. `semicolon`

Answer the following questions based on the following code snippet.

Assume you are processing a list of cats stored in `cats.csv`. The following code has been provided to you. You can assume that the `process_csv function has been declared correctly and returns a list of lists of strings where the first inner list contains the header columns and the subsequent lists contain the values for each column for one row.  The columns in cats.csv are "name", "age", and "weight".`

```
csv_data = process_csv("cats.csv")

csv_header = csv_data[0] # ["name", "age", "weight"]
csv_rows = csv_data[1:]

def cell(row_idx, col_name):
    col_idx = csv_header.index(col_name)
    val = csv_rows[row_idx][col_idx]
    return val
```

10. Assume the csv file has more than two rows. What is the output of

```
print(type(cell(1, "weight")))
```

    A. `str`
    B. `int`
    C. `list`
    D. `This code will throw an error`

11. The following function looks for the age of the oldest cat:

```
def oldest_cat():                        # LINE 1
    target_cat_age = 0                   # LINE 2
    for idx in [blank 1]:                # LINE 3
        cat_age = [blank 2]              # LINE 4
        if cat_age > target_cat_age:     # LINE 5
            target_cat_age = cat_age     # LINE 6
    return target_cat_age                # LINE 7
```

Please fill in the blanks on line 3 and line 4 to find the integer age of the oldest cat:

```
A. blank 1:  csv_rows
   blank 2:  cell(idx, "age")
B. blank 1:  csv_rows
   blank 2:  int(cell(idx, "age"))
C. blank 1:  range(len(csv_rows))
   blank 2:  cell(idx, "age")
D. blank 1:  range(len(csv_rows))
   blank 2:  int(cell(idx, "age"))
```

12. Assume that you have correctly implemented line 3 and line 4 in the `oldest_cat` function in the previous question. Now, you want to modify the function so that it instead returns the age of the youngest cat. Which of the following changes must be made to return the integer age of the youngest cat in the dataset? You can assume that there no cats in the dataset older than 20 years old.

    A. Line 5: `if cat_age < target_cat_age:`
    B. Line 2: `target_cat_age = -100`
    C. Line 2: `target_cat_age = 100`
    D. Both A and B
    E. Both A and C

## Objects & References

13. Assume that you have been provided the following code which creates a namedtuple datastructure for movies:

```
from collections import namedtuple
Movie = namedtuple("Movie", ["title", "date", "revenue"])
```

Which of the following represent a valid initialization of a movie using the provided namedtuple?

A. `Movie("Title1", "11-08")`

B. `Movie("Title1", "11-08", 1000)`

C. `Movie(Title = "Title1", Date = "11-08", Revenue = 1000)`

D. `Movie(movie)`

E. `Movie(["Title1", "11-08", 1000])`

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

```
city_list = [
    {
        "Name" : "Chicago",
        "State" : "Illinois",
        "Airports" : ["ORD", "MDW"]
    },
    {
        "Name" : "New York City",
        "State" : "New York",
        "Airports" :  ["JFK", "LGA", "EWR"]
    }
]

print(len(city_list[0]["Airports"]) - len(city_list[0]["Airports"][1]))
```

A. -2

B. 0

C. 1

D. 2

E. -1

15. The following code has been provided to you:

```
from collections import namedtuple
Student = namedtuple("Student", ["name", "grade"])

list_tup = [
    Student("Alice", 92),
    Student("Bob", 78)
]

list_dict = [
    {
        "name" : "Alice",
        "grade" : 92
    },
    {
        "name" : "Bob",
        "grade" : 77
    }
]
```

Which of the following code snippets correctly calculate the difference between Bob's grade in the list of tuples and the list of dictionaries?

A. `list_tup["Bob"]["grade"] - list_dict["Bob"]["grade"]`

B. `list_tup["1"]["grade"] - list_dict["1"]["grade"]`

C. `list_tup[1].grade - list_dict[1]["grade"]`

D. `list_tup[1].get("grade") - list_dict[1].get("grade")`

E. `list_tup[1]["grade"] - list_dict[1].grade`

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

```
lst = ["one", "two", "three"]
lst.append(4)
lst.append([7, 11])
lst.extend([55, 66, 77, 88, 99])
print(len(lst) - lst.index(4))
```

A. 4

B. 5

C. 6

D. 7

E. This code will throw an error

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

```
import copy

original_list = [1, 2, [3, 4]]
copy1 = copy.copy(original_list)
copy2 = copy.deepcopy(original_list)

copy1[2][0] = 10
copy2[2][1] = 12

final_sum = sum(original_list[2]) + sum(copy1[2]) + sum(copy2[2])
print(final_sum)
```

A. `51`

B. `36`

C. `42`

D. `43`

E. `52`

18. What is the output of the following code?

```
student_info = {
    "Alice": 22,
    16: "Bob",
    "Carol": 30
}

result = []

for var in student_info:
    result.append(var)

print(result)
```

A. `["Alice", 16, "Carol"]`

B. `["Alice", "Bob", "Carol"]`

C. `[22, 16, 30]`

D. `[22, "Bob", 30]`

E. `None of the above`

19. What is the value of `copied` after running the code below?

```
original = ["A", "B", "C", "D", "E"]
copied = original
copied[3] = "Z"
original[1] = "X"
```

A. `["A", "X", "C", "D", "E"]`

B. `["A", "X", "C", "Z", "E"]`

C. `["X", "B", "Z", "D", "E"]`

D. `["A", "B", "C", "Z", "E"]`

E. `This code will throw an error`

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

```
my_list = ["hI", 17, False, 3.14, None, ["CS", "iS", "gReAt!"]]
print(my_list[-1][-2].upper()[2])
```

A. `"I"`

B. `"A"`

C. `"a"`

D. `"E"`

E. `This code will throw an error`

21. What is the output of the following code snippet? Please select the answer with the correct type and the correct elements, the order does not matter.

```
my_list = [10, 20, 30, 30, 40, 40, 50, 10, 60, 30]
print(list(set(my_list)))
```

A. `[10, 20, 30, 30, 40, 40, 50, 10, 60, 30]`

B. `[10, 20, 30, 40, 50, 60]`

C. `{10, 20, 30, 30, 40, 40, 50, 10, 60, 30}`

D. `{10, 20, 30, 40, 50, 60}`

22. Consider the list of dictionaries below, where each dictionary contains information about one student:

```
students = [{"name": "Musa", "major": "Computer Science", "minor": None},
            {"name": "Aboli", "major": "Computer Science", "minor": "Maths"},
            {"name": "Corey", "major": "Economics", "minor": "Finance"},
            {"name": "Fahad", "major": "Data Science", "minor": "Statistics"},
            {"name": "Naima", "major": "Data Science", "minor": None}]
```

Which of the following will return a list of all the minors in `students` and if a student does not have a minor, it should return the major for that student.

A. `[student["minor"] if student["minor"] != None else student["major"] for student in students]`

B. `[student["minor"] if student["minor"] == None else student["major"] for student in students]`

C. `[student["minor"] for student in students if student["minor"] != None else student["major"]]`

D. `[student["minor"] for student in students if student["minor"] == None else student["major"]]`

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

```
def get_sum(city_tuple):
    return len(city_tuple[0]) + len(city_tuple[1])

names = [("Madison", "San Francisco"),
         ("Madison", "Chicago"),
         ("Chicago", "California")]

names.sort(key = get_sum)
print(names[-2])
```

A. `("Madison", "San Francisco")`

B. `("Madison", "Chicago")`

C. `("Chicago", "California")`

D. `Prints an empty tuple`

24. Consider the following dictionary:

```
sizes = {"medium": 2, "large": 3, "small": 1}
```

Which of the following will return a list of keys from a dictionary in the order that would correspond to the sorted values? For example, for the above dictionary, the code should output `["small", "medium", "large"]` since they would correspond to the list of sorted values `[1,2,3]`.

A. `list(sorted(sizes.items(), key = lambda k:  k[0]))`

B. `list(sorted(sizes, key = lambda k:k))`

C. `list(sorted(sizes, key = lambda k:sizes[k]))`

D. `dict(sorted(sizes.items(), key = lambda k:  k[0]))`

## Files & Errors

The following functions will be used for the next 3 questions. The code defines functions to calculate the size of pizza slices and analyze multiple pizzas. It handles exceptions for invalid input and prints the size of each slice, along with the type of exception if some types of errors occur.

```
def pizza_size(radius):
    if radius <= 0:
        raise ArithmeticError("Invalid radius size!")
    return (radius ** 2) * 3

def slice_size(radius, slice_count):
    if slice_count < 0:
        raise ArithmeticError("Invalid slice count!")
    total_size = pizza_size(radius)
    return total_size * (1 / slice_count)

def pizza_analysis(diameter_list, slices_list, pizza_count):
    for i in range(pizza_count):
        try:
            radius = float(diameter_list[i]) / 2
            slices = int(slices_list[i])
        except (ValueError, TypeError, IndexError) as e:
            print("Type of exception:", type(e))
            continue

        try:
            size = slice_size(radius, slices)
            print("slice area=" + str(size))
        except (ZeroDivisionError, ArithmeticError) as e:
            print("Type of exception:", type(e))
```

25. How many times will `Type of exception: <class "IndexError">` be printed out?

```
diameter_list = [10, "1"]
slices_list = [5, 6]
pizza_analysis(diameter_list, slices_list, 4)
```

A. `1`

B. `4`

C. `2`

D. `3`

E. `This code will not run due to an uncaught error`

26. What will be the last line of printed output from the program?

```
diameter_list = {10: "1"}
slices_list = [5, 6]
pizza_analysis(diameter_list, slices_list, 4)
```

A. `Type of exception: <class "IndexError">`

B. `Type of exception: <class "KeyError">`

C. `Type of exception: <class "TypeError">`

D. `This code will not run due to an uncaught error`

27. What will be the last line of printed output from the program?

```
diameter_list = [10, 0]
slices_list = [5, 5]
pizza_analysis(diameter_list, slices_list, 1)
```

A. `slice_area=15.0`

B. `Invalid radius size!`

C. `Type of exception: <class "ArithmeticError">`

D. `Type of exception: <class "IndexError">`

E. `This code will not run due to an uncaught error`

28. What is the output of the following code?

```
soccer_players = [
                  {"Name": "K. Mbappé", "Age": 24, "Value": 181},
                  {"Name": "J. Yates", "Age": 19, "Value": 0}
                 ]
output = []

for player in soccer_players:
    for key in player:
        if key != "Value":
            try:
                assert type(player[key]) == int
                val_ratio = player[key] / player["Value"]
                output.append("success")
            except ZeroDivisionError as e:
                output.append("0 division")
            except AssertionError as e:
                output.append("failed assert")

print(output)
```

A. `AssertionError`

B. `["failed assert", "success", "failed assert", "0 division"]`

C. `["failed assert", "success", "0 division", "success"]`

D. `ZeroDivisionError`

E. `["failed assert", "success", "failed assert", "success"]`

29. In what order should the following lines of code be executed if you want to save the contents of `old.json` to `new.json`? You can safely ignore the level of indentation for this question. You can also assume that both `old.json` and `new.json` are valid JSON files in the current directory.

```
1. data = json.load(f_old)
2. f_new.close()
3. f_old = open("old.json", encoding="utf-8"):
4. json.dump(data, f_new)
5. f_new = open("new.json", "w", encoding="utf-8"):
6. import json
7. f_old.close()
```

A. `6, 5, 3, 4, 1, 7, 2`

B. `6, 5, 3, 1, 4, 2, 7`

C. `6, 3, 5, 4, 1, 7, 2`

D. `3, 5, 6, 7, 2, 1, 4`

E. `1, 2, 3, 4, 5, 6, 7`

30. What will be the content of the `test_file.txt` file after executing the following code snippet? The file did not exist previously.

```
f = open("test_file.txt", "w")
f.write("Hello\n")
f.close()

f = open("test_file.txt", "w")
f.write("Hello")
f.write("World!")
f.close()
```

A.
```
Hello
HelloWorld!
```

B. `HelloWorld!`

C. `Hello World!`

D.
```
Hello
Hello
World!
```

E.
```
Hello
World!
```

(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)