CS 220 - Spring 2025

Instructors: Mike Doescher and Louis Oliphant

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 09:55 AM (Louis)
   004 - MWF 01:20 PM (Louis)
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, note sheet, and filled-in scantron form. The note sheet will not be returned.

General

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

```
s1 = "Bucky Badger"
s2 = "Loves CS220!"
s3 = s1 + " , " + s2
print(s3[:6] + s3[-6:-1])
```

   A. "BuckyCS220"

   B. "Bucky CS220"

   C. "BuckyCS220!"

   D. "Bucky CS220!"

   E. "Bucky S220!"

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

```
msg = "I Love CS220!"
new_msg = ""
i = 2
while  i < len(msg):
    new_msg += msg[i].upper()
    i+= 2
print(new_msg)
```

   A. "LV S2!"

   B. "ILV S2!"

   C. "ILVS2!"

   D. "LV S2"

   E. None of the above

3. What is the output of the following code?

```
rows = [["State", "City", "Population 2023 (M)", "Population 2024 (M)"],
        ["Illinois", "Chicago", 2.6, 2.7 ],
        ["Michigan", "Detroit", 0.6, 0.6],
        ["Wisconsin", "Milwaukee", 0.5, 0.6 ],
        ["Indiana", "Indianapolis", 0.8, 0.9 ] ]

rows = rows[1:]

count = 0
for i in rows:
    if len(i[1]) <= 7:
        count+= i[-2]

print(count)
```

A. `0`

B. `3.1`

C. `3.3`

D. `3.2`

E. `None of the above`

4. What is the output of the following code?

```
a = [4, 1]
b = [2, 3]
c = [5, 8]
b.extend(a)
c.pop(-1)
print(b, c)
```

A. `[2, 3, [4,1]] [5]`

B. `[[2, 3], [4,1]] [5,8]`

C. `[2, 3, 4, 1] [5]`

D. `[2, 3] [5, 8]`

E. `[2, 3, 4, 1, 5]`

5. What is the output of the following code?

```
fruits = ["Pear", "Orange", "Apple", "Banana" ]
new_list = []

for fruit in fruits:
    if fruit[0] <= "O":
        new_list.append(fruit)
new_list.sort(reverse = True)
print(new_list)
```

   A. `["Orange", "Banana", "Apple"]`
   B. `["Pear", "Orange", "Apple", "Banana"]`
   C. `["Pear", "Orange"]`
   D. `["Apple", "Banana", "Orange"]`
   E. `["Orange", "Pear"]`

6. Which of the following will access the value 18, the number of games Federer won?

```
player = {}
player["name"] = {1: "Federer"}
player["stats"] = {"Winners": 31, "Games Won": 18, "Double Faults": 0}
```

   A. `player["Federer"]["Games Won"]`
   B. `player["Games Won"]`
   C. `stats["Federer"]["Games Won"]`
   D. `player["Games Won"]["stats"]`
   E. `player["stats"]["Games Won"]`

7. What is the output of the following code?

```
my_dict = {0: "Mike", 2: "Bob"}
my_dict[1] = "Louis"
my_dict.pop(2)
print(list(my_dict.values()))
```

A. `["Mike" "Bob"]`

B. `[0, 2, 1]`

C. `["Mike" "Bob", "Louis"]`

D. `["Mike", "Louis"]`

E. `[0, 1]`

8. A student is analyzing a dataset stored in the CSV file `survey.csv`. The dataset has the following structure:

```
Name, Age, Major, Procrastinator
Alice, 19, Computer Science, Yes
Bob, 20, Mathematics, No
Charlie, 21, Physics, Yes
Daisy, 22, Engineering, No
```

Given that the `process_csv()` function reads the contents of a CSV file and returns it as a list of lists, where each inner list represents a row in the CSV file, what will the following code output?

```
survey_data = process_csv("survey.csv")
headers = survey_data[0]
data = survey_data[1:]

for row in data:
    if "Procrastinator" in headers:
        idx = headers.index("Procrastinator")
        if row[idx] == "Yes":
            print(row[headers.index("Major")])
```

A. It prints the `"Major"` of only the first student who answered `"Yes"` in the `"Procrastinator"` column.

B. It will output/throw an error.

C. It prints nothing because the condition of the if-statement `"Procrastinator" in headers` is false.

D. It prints the `"Major"` of every student who answered `"Yes"` in the `"Procrastinator"` column.

E. It prints the `"Major"` of every student.

9. Consider the following Python code:

```
from collections import namedtuple

Point = namedtuple("Point", ["x", "y"])
p1 = Point(3, 4)

t1 = (5, 6)
t2 = t1
p2 = p1

t1 += (7, 8)
p1 = Point(10, 20)

print(t1)
print(t2)
print(p1)
print(p2)
```

What will be printed?

A. (5, 6, 7, 8)
(5, 6, 7, 8)
Point(x=10, y=20)
Point(x=3, y=4)

B. (5, 6, 7, 8)
(5, 6, 7, 8)
Point(x=10, y=20)
Point(x=10, y=20)

C. (5, 6, 7, 8)
(5, 6)
Point(x=3, y=4)
Point(x=10, y=20)

D. (5, 6, 7, 8)
(5, 6)
Point(x=3, y=4)
Point(x=3, y=4)

E. (5, 6, 7, 8)
(5, 6)
Point(x=10, y=20)
Point(x=3, y=4)

10. Consider the following data stored as a list of lists:

```
company_data = [
    ["Department", "Employee", "Age", "Salary", "Projects"],
    ["Engineering", "Alice", 29, 120000, ["AI", "Cloud"]],
    ["Engineering", "Bob", 35, 135000, ["Security", "DevOps"]],
    ["Marketing", "Charlie", 30, 90000, ["SEO", "Branding"]],
    ["Marketing", "David", 40, 110000, ["Social Media", "Ads"]]
]
```

The following code converts the above dataset into a dictionary:

```
def convert_data_to_dictionary(data):
    header = data[0]
    output_dict = {}
    for i in range(1,len(data)):
        inner_list = data[i]
        if inner_list[0] not in output_dict:
            output_dict[inner_list[0]]={}
        output_dict[inner_list[0]][inner_list[1]] = {
            header[2]:inner_list[2],
            header[3]:inner_list[3],
            header[4]:inner_list[4]}
    return output_dict

company_data_dict =  convert_data_to_dictionary(company_data)
```

Which of the following would correctly return Alice's Salary?

A. `company_data_dict["Engineering"]["Alice"]["Salary"]`

B. `company_data_dict["Alice"]["salary"]`

C. `company_data_dict["Engineering"]["Alice"][1]`

D. `company_data_dict["Engineering"]["Alice"][3]`

E. None of the Above

11. Given the company_data_dict from the above question, the following function tries to print the age of all employees. What should be replaced in the `???` placeholders, respectively, so that the function will correctly print the age of each employee?

```
for i in company_data_dict:
    for j in ???:
        print("Age:", ???)
```

A. `i, j["Age"]`

B. `company_data_dict[i], j["Age"]`

C. `i, company_data_dict[i][j]["Age"]`

D. `company_data_dict[i], company_data_dict[i][j]["Age"]`

E. `None of the above`

12. What will be the output of the following code?

```
import copy
a = [1,2,[3,4,5]]
b = copy.copy(a)
b[2].append(3)
b.append(4)
print(a)
```

A. `[1, 2, [3, 4, 5]]`

B. `[1, 2, [3, 4, 5, 3]]`

C. `[1, 2, [3, 4, 5], 4]`

D. `[1, 2, [3, 4, 5, 3], 4]`

E. `None of the above`

13. What will be the output of the following code?

```
def func(n):
    if n <= 0:
        return 1
    return n + func(n - 1) * func(n - 2)

print(func(3))
```

A. `4`

B. `5`

C. `11`

D. `12`

E. `None of the Above`

14. Which of the following lines of code makes a new reference to the **get_mean** function object? Choose all options that are correct

```
def get_mean(items):
    sum_of_items = 0
    for i in range(len(items)):
        sum_of_items += items[i]
    mean_of_items = sum_of_items / len(items)
    return mean_of_items

I)   func = get_mean
II)  func = get_mean()
III) func = get_mean([])
IV)  func = get_mean([1, 2, 3])
```

A. All four of the choices

B. III and IV only

C. I and II only

D. I only

E. None of the choices

15. What is the output of the following code?

```
numbers = [1, 2, 3, 4, 5, 6]
result = [x ** 2 for x in numbers if x % 2 == 1]
print(result)
```

A. [1, 4, 9, 16, 25, 36]

B. [1, 9, 25]

C. [4, 16, 36]

D. [2, 6, 10]

E. [4, 8, 12]

16. Consider the list of dictionaries, where each dictionary contains the information for one video game.

```
games = [{
         "name": "The Legend of Zelda: Breath of the Wild",
         "release_year": 2017,
         "rating": 97,
         "console": "Nintendo Switch"
         },
         {
         "name": "Super Mario Odyssey",
         "release_year": 2017,
         "rating": 97,
         "console": "Nintendo Switch"
         },
         {
         "name": "Minecraft",
         "release_year": 2011,
         "rating": 93,
         "console": "PC"
         },
         {
         "name": "Sonic Mania",
         "release_year": 2017,
         "rating": 86,
         "console": "PlayStation 4"
         },
         {
         "name": "Splatoon 3",
         "release_year": 2022,
         "rating": 83,
         "console": "Nintendo Switch"
         }
]
```

Which of the following lines of code make a list of video game **names** for **console** type **Nintendo Switch** and **rating** of at least **90**?

A. `[game["name"] for game in games if game["console"] == "Nintendo Switch"]`

B. `[game["name"] for game in games if game["rating"] >= 90]`

C. `[game for game in games if game["console"] == "Nintendo Switch" and game["rating"] >= 90]`

D. `[game["name"] for game in games if game["console"] == "Nintendo Switch" and game["rating"] >= 90]`

E. `[game["name"] for game in games if "Nintendo Switch" in game]`

17. Consider the following code:

```
exam_questions = [
    {
    "question": 1,
    "is_tricky" : False,
    "exam" : 1
    },
    {
    "question": 2,
    "is_tricky": True,
    "exam": 1
    }
]

target_question = 2
for i in range(len(exam_questions)):
    if exam_questions["question"] == target_question:
        print(exam_questions["is_tricky"], end=" ")
print("CS220")
```

What is the output after running the above code?

A. True

B. False CS220

C. True CS220

D. The code will raise an error

E. CS220

18. What text will be present in **file1.txt** after this code runs?

```
f1 = open("file1.txt", "w")
f1.write("HelloWorld")
f1.close()

f1 = open("file1.txt", "r")
f2 = open("file2.txt", "w")
f2.write(f1.read())
f1.close()
f2.close()

f1 = open("file1.txt", "w")
f1.write("CS220")
f1.close()
```

What text will be present in **file1.txt** after this code runs?

A. HelloWorld

B. CS220

C. This code will raise an error

D. HelloWorldCS220

E. None of the above

Power Generators

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

```
data = [
    ["plant_id", "plant_name", "summer_capacity", "winter_capacity", "technology"],
    [1756, "Saxon Falls", 1.3, 2.2, "Conventional Hydroelectric"],
    [3974, "Wisconsin Rapids", 0.6, 3.6, "Conventional Hydroelectric"],
    [3991, "Fitchburg", 15.8, 16.2, "Natural Gas Fired Combustion Turbine"],
    [4020, "Frederic Diesel", 0.7, 0.8, "Petroleum Liquids"],
    [3452, "Saxon Falls", 0.7, 3.1, "Natural Gas Fired Combustion Turbine"]
]
```

19. Which of the following lines of code returns the **maximum total capacity** (sum of summer and winter capacity) of the plant with the name "**Saxon Falls**"?

    A. `max([row[2] + row[3] for row in data[1:]])`

    B. `max([row[2] for row in data[1:]  if row[1] == "Saxon Falls"])`

    C. `[row[2] + row[3] for row in data[1:]  if row[1] == "Saxon Falls"]`

    D. `[row[2] + row[3] for row in data[1:]]`

    E. `max([row[2] + row[3] for row in data[1:]  if row[1] == "Saxon Falls"])`

20. What is the output of the following code?

```
print(data[0].index("net_summer_capacity") +
        data[2][data.index("net_winter_capacity")])
```

    A. 5.6

    B. 6.6

    C. 2.6

    D. 4.2

    E. This code throws an error

21. What is the output after the following code runs?

```
results = []
header = data[0]
for row in data[1:]:
    diff = row[header.index("winter_capacity")] -
                row[header.index("summer_capacity")]
    if diff < 1.0:
        results.append(row[header.index("plant_id")])
sorted(results, reverse=True)
print(results)
```

A. [1756, 3452]

B. [3974, 3452]

C. [3452, 3974]

D. [1756, 3991, 4020]

E. [4020, 3991, 1756]

Movies

Assume you have run the following code snippet from P8

```
 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

csv_data = process_csv("movies.csv")
csv_rows = csv_data[1:]
csv_header = csv_data[0]

print(csv_header)
#Output: ["title", "year", "duration", "genres", "rating", "directors", "cast"]
```

Same as P8, csv_rows contains the data in a list of lists. To make it easier to access the data, we then convert it to a list of dictionaries named `raw_movies_list`. The data structure looks like this:

```
[
 {"title": "tt0115953",
  "year": "1996",
  "duration": "90",
  "genres": "Crime, Mystery, Thriller",
  "rating": "3.9",
  "directors": "nm0915394",
  "cast": "nm0023858, nm0001614, nm0518715, nm0001457"},
 {"title": "tt0283453",
  "year": "2003",
  "duration": "95",
  "genres": "Drama, Romance, Thriller",
  "rating": "5.5",
  "directors": "nm0557751",
  "cast": "nm0000117, nm0121605, nm0000998, nm0694052"},
 {"title": "tt0447619",
  "year": "2006",
  "duration": "102",
  "genres": "Adventure, Crime, Drama",
  "rating": "4.8",
  "directors": "nm0630112",
  "cast": "nm0000899, nm0000366, nm0001953, nm0098793"}
  ...
]
```

22. Now we want to convert the above raw_movies_list into a list of dictionaries with the field `"cast"` converted to a list of strings. For example, we want to convert `"nm0000899, nm0000366, nm0001953, nm0098793"` into `["nm0000899", "nm0000366", "nm0001953", "nm0098793"]`.
Note: in the original string, the `cast` are separated by a comma followed by a space `", "` and we would like to remove them in the converted list.

```
for movie in raw_movies_list:
    movie["cast"] =  ...
```

Which of the following should replace the ... to correctly modify `raw_movies_list`?

A. `movie["cast"].split(",")`

B. `movie["cast"].replace(" ","").split(",")`

C. `list(movie["cast"])`

D. `movie["cast"].tolist()`

E. `None of the Above`

23. Similar to P8, you also have a `mapping.csv` file that stores mappings of `"IMDb ID"`'s of movies, cast members, and directors to their corresponding name. Suppose you run the following code:

```
mapping_rows = process_csv("mapping.csv")
print(mapping_rows[:10])
```

```
#Output:
[["tt7708470", "The Bloody Man"],
 ["tt5759068", "Freightened: The Real Price of Shipping"],
 ["tt0081541", "The Under-Gifted"],
 ["tt1024962", "Here We Are!"],
 ["tt0246875", "Purely Belter"],
 ["tt0239341", "Devi Putrudu"],
 ["tt0360960", "Shadows of Time"],
 ["tt3215486", "Idhu Kathirvelan Kadhal"],
 ["tt12792532", "The Third War"],
 ["tt0070660", "Seduction"]]
```

Now you have the data in `mapping.csv` stored as a list of lists. Same as in P8, we want to convert it into a dictionary where the key is the ID and value is the corresponding name. Which of the following code should we use in place of the `...` in the function to correctly construct the dictionary?

```
def convert_rows_to_dicts(mapping_rows):
    mapping_dict = {}
    for row in mapping_rows:
        ...

    return mapping_dict

convert_rows_to_dicts(mapping_rows)
```

A. `mapping_dict[row[0]] = row[1]`

B. `mapping_dict[row[0]].append(row[1])`

C. `{row[0]:row[1] for row in mapping_rows}`

D. `mapping_dict.update(row[1]:  row[0])`

E. None of the Above

24. Now we can convert each cast's ID in `raw_movies_list` into their actual name using the `mapping_dict` we created. Choose the option that completes the following code snippet so that the IDs of the casts are converted into their actual name in `mapping.csv`.

```
for movie in raw_movies_list:
    ...
```

A. `movie["cast"] = [mapping_dict[i] for i in movie["cast"]]`

B. `movie["cast"] = [mapping_dict[i] for i in range(len(movie["cast"]))]`

C. `movie["cast"] = [mapping_dict.append(i) for i in movie["cast"]]`

D. `movie["cast"] = mapping_dict[movie["cast"]]`

E. `None of the Above`

Steam Games

The following functions are the same as in P5.

- `project.get_name(idx)`: The name of the game at row `idx`
- `project.get_publisher(idx)`: The publisher of the game in row `idx`
- `project.get_release_date(idx)`: The release date of the game in row `idx` in mm/dd/yyyy format
- `project.get_avg_playtime(idx)`: The average playtime (in hours) of the game at row `idx`
- `project.get_price(idx)`: The price of the game at row `idx`, like $19.99
- `project.get_positive_reviews(idx)`: The number of positive reviews for the game at row `idx`, like 2.81K
- `project.get_negative_reviews(idx)`: The number of negative reviews for the game at row `idx`, like 1.13K
- `project.count()`: The total number of games

As in P5, the dataset has the columns `name`, `publisher`, `release_date`, `avg_playtime`, `price`, `positive_reviews`, and `negative_reviews`.

25. What should the `???` be replaced with to find the highest price of a game published by "Valve"? Assume that the `format_price` function is correctly defined and returns the price of a game as a float.

```
import project

highest_valve_price = None
for idx in range(project.count()):
   if ???:
        continue
   price = format_price(project.get_price(idx))
   if highest_valve_price == None or price >= highest_valve_price:
        highest_valve_price = price
```

A. `project.get_publisher(idx) == "Valve"`

B. `project.get_publisher(idx) != "Valve"`

C. `project.get_publisher(idx) >= "Valve"`

D. `project.get_publisher(idx) <= "Valve"`

E. `None of the above`

Use the following function definition and dataset to answer the next two questions. The functionality of `format_num_reviews` is exactly the same as in P5.

```
def format_num_reviews(num_reviews):
    last_char = num_reviews[-1]
    if last_char == "M":
        review_num = float(num_reviews[:-1]) * 1e6
    elif last_char == "K":
        review_num = float(num_reviews[:-1]) * 1e3
    else:
        review_num = float(num_reviews)
    return review_num
```

Name	Release Date	Avg Playtime	Price	Pos. Reviews	Neg. Reviews
DOOM Eternal	03/19/2020	456	$39.99	165.67K	16.43K
Just Cause 2	03/23/2010	633	$14.99	45.264K	4.57K
War Thunder	08/15/2013	408	$0	353.04K	227.01K
Half-Life 2	11/16/2004	704	$9.99	176.63K	4.37K
Alan Wake	02/16/2012	207	$14.99	38.67K	3.95K

26. What will the output of the following code be?

```
import project

highest_val = None
game_of_interest = None
for idx in range(project.count()):
    current = format_num_reviews(project.get_positive_reviews(idx)) -
        format_num_reviews(project.get_negative_reviews(idx))
    if highest_val == None or current >= highest_val:
        highest_val = current
        game_of_interest = project.get_name(idx)
print(game_of_interest)
```

A. DOOM Eternal
B. Just Cause 2
C. War Thunder
D. Half-Life 2
E. Alan Wake

27. What will the output of the following code be?

```
import project

highest_val = None
game_of_interest = None
for idx in range(project.count()):
    current = int(project.get_release_date(idx)[-4:])%4
    if highest_val == None or current >= highest_val:
        highest_val = current
        game_of_interest = project.get_name(idx)
print(game_of_interest)
```

A. DOOM Eternal
B. Just Cause 2
C. War Thunder
D. Half-Life 2
E. Alan Wake

Soccer Stars

In Project 7, the dictionary `player_dict_by_id` is used to store player data, where each key is a player's `"ID"` and the value is another dictionary containing the player's attributes (e.g., `"Name"`, `"Age"`, `"Nationality"`, etc.). Specifically, the first few key/value pairs of `player_dict_by_id` look like this:

```
player_dict_by_id = {
239085: {"ID": 239085,
          "Name": "E. Haaland",
          "Age": 22,
          "Nationality": "Norway",
          "Team": "Manchester City",
          "League": "Premier League (England)",
          ...},
 231747: {"ID": 231747,
          "Name": "K. Mbappé",
          "Age": 24,
          "Nationality": "France",
          "Team": "Paris Saint Germain",
          "League": "Ligue 1 (France)",
          ...},
...}
```

28. You are tasked with creating a function `get_players_by_stat` that takes three arguments: `player_dict_by_id`, a `stat` (e.g., `"Attacking"`, `"Movement"`), and a `threshold` value. The function should return a dictionary where each key is a player's `ID` and the value is their `"Name"`, but only for players whose `stat` is greater than or equal to the `threshold`.
What is correct to fill in all of the `???` placeholders, respectively, in the following function?

```
def get_players_by_stat(player_dict_by_id, stat, threshold):
    result = {}
    for player_id in player_dict_by_id:
        if player_dict_by_id[player_id][stat] ??? threshold:
            result[???] = ???
    return result
```

A. `>`, `player_id`, `player_dict_by_id[player_id]["Name"]`

B. `>=`, `player_dict_by_id`, `player_dict_by_id[player_id][stat]`

C. `>`, `player_id`, `player_dict_by_id[player_id][stat]`

D. `>=`, `player_id`, `player_dict_by_id[player_id]["Name"]`

E. `>=`, `player_dict_by_id["Name"]`, `player_id`

29. Assume you have created the function `get_players_by_stat` from the question above. Now, you want to find the **number of players** whose `"Attacking"` stat is greater than or equal to `80`. What is correct to fill in the `???` to calculate this?

```
players_by_attacking = get_players_by_stat(player_dict_by_id, "Attacking", 80)
num_players = ???
```

A. `sum(players_by_attacking.values())`

B. `len(players_by_attacking)`

C. `players_by_attacking.count()`

D. `players_by_attacking.keys()`

E. `players_by_attacking.values()`

30. You are given the dictionary `player_dict_by_id` and the dictionary `players_by_attacking` (from the question above). Now, you want to create a new dictionary `players_by_team_and_attacking` that maps each `"Team"` to a list of player names who have an `"Attacking"` stat greater than or equal to `80`. Which of the following code snippets correctly **continue the given code** by filling in the lines to implement this?

```
players_by_team_and_attacking = {}
for player_id in players_by_attacking:
    team = player_dict_by_id[player_id]["Team"]
    ####Fill in to complete####
```

A.
```
if team not in players_by_team_and_attacking:
        players_by_team_and_attacking[team] = []
players_by_team_and_attacking[team].append(
player_dict_by_id[player_id]["Name"])
```

B.
```
players_by_team_and_attacking[team] = players_by_attacking[player_id]
```

C.
```
if team in players_by_team_and_attacking:
        players_by_team_and_attacking[team].append(
        players_by_attacking[player_id])
```

D.
```
players_by_team_and_attacking[team] = []
players_by_team_and_attacking[team].append(players_by_attacking[player_id])
```

Blank Page: this page is intentionally blank