CSCI 1470 CS 1
Spring 2026
Final Examination

Last Name: _____ **First Name:** _____ **Student Id:** _____

Number: _____

Time allowed: *2 hours.* Total score: 100 points. *Closed* book examination. *Two letter-size information sheets (both sides) prepared by yourself are allowed.* This question paper is printed on *both sides*.

Answer all questions. Turn in everything: question and answer papers, information sheets and sketch papers. They will be stapled together.

[Q1] (22%) Multiple Choices. Please circle a choice as your answer. Alternatively, you can write your choice clearly.

[1] Which choice fills in the blank so that the output prints one line for each item in sports_list, as in: 1. Hockey?

```
sports_list = [ "Hockey", "Football", "Cricket" ]
for i in _____:
    print(f"{i+1}. {sports_list[i]}")
```

A. `range(len(sports_list))`
B. `range(len(sports_list-1)`
C. `range(1, len(sports_list))`
D. `range(1, len(sports_list)-1)`

[2] What is the output?

```
x = 18
while x % 3 == 0:
    print(x, end=" ")
    x = x // 3
```

A. 6
B. 6 2
C. 18 6 2
D. 18 6

[3] How many times will the print statement execute?

```
for i in range(10):
    for j in range(3):
        print(f"{i}. {j}")
```

A. 3
B. 10
C. 13
D. 30

[4] After the program runs, what is the value of y?

```
def print_sum(num1, num2)
    print(num1 + num2)
y = print_sum(4, 5)
```

A. 4 5
B. 9
C. 45
D. None

[5] Which of the following is not a reason to use functions?

A. To avoid writing redundant code
B. To improve code readability
C. To support modular development
D. To make the code run faster

[6] Which of the following lines of code is not valid, given the definitions of the calc_cube() and display() functions?

```
def calc_cube(x):
    return x * x * x

def display(x):
    print(x)
```

A. `y = calc_cube(2.0)`
B. `calc_cube(x) = 8.0`
C. `display("Test")`
D. `display(calc_cube(2.0))`

[7] What is the output of running this program?

```
def is_even(num):
    if num % 2 == 0:
        even = True
    else:
        even = False

is_even(7)
print(even)
```

A. False
B. True
C. No output: Call to is_even() fails due to no return value
D. No output: An error occurs due to unknown variable even

[8] What is the output?

```
my_poem = "Roses are red; Violets are blue"
new_separator = "."
new_poem = my_poem.split(";")
print(new_separator.join(new_poem))
```

A. Roses are red. Violets are blue
B. Roses.are.red.Violets.are.blue
C. Roses;are;red;Violets;are;blue
D. Roses are red Violets are blue

[9] Given a list `my_list = [[0, 1, 2], [3, 4, 5], [6, 7, 8]]`, how would you access the value 7?

A. `my_list[2][1]`
B. `my_list[2][2]`
C. `my_list[3][2]`
D. `my_list[3][1]`

[10] What is output?

```
objects = {}
objects["a"] = "Chair"
objects["b"] = "Table"
objects["c"] = "Sofa"
objects.clear()
print(objects)
```

A. {'a':'Chair', 'b':'Table', 'c':'Sofa'}
B. {'Chair', 'Table', 'Sofa'}
C. {}
D. {'a', 'b', 'c'}

[11] Which of the following statements correctly opens a file in append mode without the need for specifying a close() function for the file?

A. `with open("my_list.txt", "a+") as file:`
B. `file = with open("my_list.txt", "a+")`
C. `with open("my_list.txt", "rb") as file:`
D. `file = with open("my_list.txt", a)`

[Q2] (15%) True or False (Circle one choice, or write either True or False)

[a] [T F] The break statement in Python is a control flow statement used within a loop to skip the current iteration of the loop.

[b] [T F] In Python, a function can return three integers.

[c] [T F] In Python, FileNotFoundError is an example of syntax errors.

[d] [T F] In Python, the file mode `"r"` is used for reading a file.

[e] [T F] In Python, the key of a dict can be a list.

[f] [T F] Python's function supports named parameters and arguments.

(g) [T F] The numpy library is a built-in standard library in Python.

(h) [T F] The matplotlib library in Python is not used to directly solve partial differential equations.

(i) [T F] In Python, a function can be passed as an argument of another function.

(j) [T F] The pandas package in Python can be used to manage pandas as well as other large mammals.

[Q3] (14%) Strings, lists and slices

[a] (5%) What is the output of executing the following code?

```
s = "123456789"
print(s[1])
print(s[3:6])
list_1 = list(s)
print(list_1[0:6:3])
print(s[::-1])
print(list_1[3:0:-1])
```

[b] (9%) What is the output of executing the following code?

```
list_2 = [10, [3,5], [2,4], 7]
print(len(list_2))
print(list_2[1])
print(list_2[2][1])
print(list_2[1:3])
print(list_2[1:3][1][1])

list_3 = list_2
list_4 = list_2[:]
list_2[2] = 99
print(list_2[2])
list_2.pop()
print(list_2)
print(list_3)
print(list_4)
```

[Q4] (30%)

[1] (8%) Write Python code to count the number of integers and the number of strings in a list of mixed data types. Output the result in the following manner. For example, consider:

```
list_1 = [1,'hello',3.14,2,'world',3,None]
```

Your code should output:

```
list_1: [1, 'hello', 3.14, 2, 'world', 3, None]
Number of integers in list_1: 3
Number of strings in list_1: 2
```

Hints: You may use the function isinstance(object, classinfo):

object: The object or variable you want to check the type of.

classinfo: A class, type, or a tuple of classes/types to check against.

For example: running the code:

```
i = 12
s = 'hello'
print(f"isinstance(i, int): {isinstance(i, int)}")
print(f"isinstance(i, str): {isinstance(i, str)}")
print(f"isinstance(s, int): {isinstance(s, int)}")
print(f"isinstance(s, str): {isinstance(s, str)}")
```

output:

```
isinstance(i, int): True
isinstance(i, str): False
isinstance(s, int): False
isinstance(2, str): True
```

[2] (8%) Write Python code to count the total number of words in a list of strings, `list_1`. Every string contains alphabets and white spaces only. A word is composed of alphabets only, separated by white spaces. For example, consider:

```
list_1 = ["hello world", "It is me", "from Houston"]
```

Your code should output:

```
list_1: ['hello world', 'It is me', 'from Houston']
Number of words: 7
```

Hints: You may consider using the split() function, which splits a string by white spaces by default. For example: executing:

```
"I am from Houston".split()
```

returns

```
['I', 'am', 'from', 'Houston']
```

[3] (7%) Write Python code to create a new dict, `dict_2`, from another dict, `dict_1`. The values of `dict_1`become the keys of `dict_2` and the keys of `dict_1` become the values of `dict_2`. You may assume that there is no duplicate values in `dict_1`. For example,

```
dict_1 = {'California': 'CA', 'Texs': 'TX', 'Alabama': 'AL'}
print(f"dict_1: {dict_1}")

# Your code to create dict_2

print(f"dict_2: {dict_2}")
```

output

```
dict_1: {'California': 'CA', 'Texs': 'TX', 'Alabama': 'AL'}
dict_2: {'CA': 'California', 'TX': 'Texs', 'AL': 'Alabama'}
```

[4] (7%) Write Python code to use a list of list of integers, `list_1`, to create another list, `list_2`. Each element of `list_2` contains the sum of the integers in the corresponding sub-lists in `list_1`. For example:

```
list_1 = [[1,2,3],[2,3],[5,6,7],[8],[]]
print(f"list_1: {list_1}")

# Your code to create list_2

print(f"list_2: {list_2}")
```

output

```
list_1: [[1, 2, 3], [2, 3], [5, 6, 7], [8], []]
list_2: [6, 5, 18, 8, 0]
```

[Q5] (20%) functions

[1] (10%) Write a function `add_number_lists(list_1, list_2)` to return a list that is a sum of the two lists. If the lists are of different lengths, the resulting list will have the length of the shorter list. For example, executing the following code:

```
list_1 = [1,2,3,4]
list_2 = [6,7,8]
print(f"list_1: {list_1}")
print(f"list_2: {list_2}")
print(f"add_number_lists(list_1, list_2): {add_number_lists(list_1,
list_2)}")
```

output:

```
list_1: [1, 2, 3, 4]
list_2: [6, 7, 8]
add_number_lists(list_1, list_2): [7, 9, 11]
```

[2] (10%) write a python function `merge_and_create_list(list_1, list_2)` to accept two lists, `list_1` and `list_2` and return a list containing elements in `list_1` and then elements in `list_2`, with all duplicate elements removed. For example, executing the following code,

```
list_1 = [1,2,8,1,2]
list_2 = [3,4,1,8]
print(f"list_1: {list_1}")
print(f"list_2: {list_2}")
print(f"merge_and_create_list(list_1, list_2):
{merge_and_create_list(list_1, list_2)}")
```

output:

```
list_1: [1, 2, 8, 1, 2]
list_2: [3, 4, 1, 8]
merge_and_create_list(list_1, list_2): [1, 2, 8, 3, 4]
```