UNIVERSITY OF IBRA
Department of Numeracy, Computation, and Probability

CSC108H5 F– (Not) Penultimate Exam
Introduction to Computer Programming

Instructors: Themba Dube, Ibrahim Chehab

Duration: 1 hour
Aids Allowed: God Himself
2023/11/30

Name: ______________________________
Student Number: ________________________

The University of Ibra and you, as a student, share a commitment to academic integrity. You are reminded that you may be charged with an academic offence for possessing any unauthorized aids during the writing of an exam. Clear, sealable, faraday bags have been provided for all mythical devices, including but not limited to: telepathic communication headsets, time machines, polygraph machines, magic wands, Miraculouses, and any other supernatural or futuristic devices that could potentially aid in unfair advantages during examinations. Please turn off all devices, seal them in the bag provided, and place the bag under your desk for the duration of the examination. You will not be able to touch the bag or its contents until the exam is over. If, during an exam, any of these items are found on your person or in the area of your desk other than in the clear, sealable, faraday bag, you may be charged with an academic offence. A typical penalty for an academic offence may cause you to do the hokey-pokey uncontrollably.
*Please note, once this exam has begun, you **CANNOT** re-write it.*

This exam contains 10 pages (including this cover page) and 9 questions. Please ensure all pages are present before starting this final examination.

Part I: Multiple Choice

Answer each question to the best of your abilities. Each question has exactly one answer.

1. (2 points) **Python Data Structures**
   Which of the following is **not** a valid type in Python?

   A. `type`

   B. `bytes`

   C. `Set`

   D. `set`

   E. `None of the above`

2. (2 points) **Code Tracing I**
   Consider the following Python function:

```
def cursed_funct_junior(i : int) -> int:
    lst = [0, 0, 1]
    for _ in range(i):
        lst.append(lst[-2])
        lst.append(lst[-2])
        lst.append(lst[-2] + lst[-1])
    return lst[-1]
```

   What is the value of `cursed_funct_junior(3)`?

   A. 0

   B. 1

   C. 2

   D. 3

3. (2 points) **Code Tracing II**
   Consider the following Python function:

```
 def cursed_funct_1(a: callable, b: callable, c: int, d:int) -> int
:
     if c > d:
         increment = lambda x: 2*x
         return a(c//2) + b(d//2)
     else:
         increment = lambda x: 4*x
         return a(c//2) - b(d//2)

 def increment(x: int) -> int:
     return x + 1

 def decrement(x: int) -> int:
     return x - 1

 def cursed_funct_2(a: callable, b: callable, c: int, d: int):
     a = cursed_funct_1 if a else increment
     b = b if a else a
     return a(b, decrement, c if a else c//2, d)


 print(cursed_funct_2(increment, decrement, 7, 10))
```

What is the output of this code?

A. -9

B. -2

C. 2

D. An `Exception` of some kind

E. None of the above

4. (2 points) **Code Tracing III**
Consider the following Python function which operates on a list:

```
def cursed_list_1(a: list, b: list) -> list:
    if len(a) > len(b):
        return [x for x in a if x not in b]
    elif len(a) == len(b):
        return [x for x in b if x not in a]
    return []

def cursed_list_2(a: list, b: list) -> list:
    a = cursed_list_1 if a else [x for x in a if x in b]
    b = b if a else a
    return a(b, [x for x in a if x not in b])

print(cursed_list_2([1, 2, 3][::-1], [2, 3, 4][:2:-3]))
```

What is the output of this code?

A. `[1, 2, 3]`

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

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

D. An `Exception` of some kind is raised

E. None of the above

5. (4 points) **Code Tracing IV**
`Ibra.java` works on a startup called TTBTrackr. Unfortunately, his code was leaked by a

rogue employee, ibra.himo. Fortunately for IbraSoft™, all their code is obfuscated. Consider the following Python method extracted from the leaked code:

```
def mystery(arr: list[int]):
    n = len(arr)
    size = 1
    while size < n:
        for left in range(0, n - 1, 2 * size):
            mid = min(left + size - 1, n - 1)
            right = min(left + 2 * size - 1, n - 1)

            scooby_doo(arr, left, mid, right)
        size *= 2


def scooby_doo(arr: list[int], a: int, b: int, c: int):
    i = a
    j = b + 1

    while i <= b and j <= c:
        if arr[i] <= arr[j]:
            i += 1
        else:
            temp = arr[j]
            for k in range(j, i, -1):
                arr[k] = arr[k - 1]
            arr[i] = temp

            i += 1
            b += 1
            j += 1

    while j <= c:
        arr[b + 1] = arr[j]
        j += 1
        b += 1
```

Question continued on next page

(a) (2 points) Is this a mutating or non-mutating method?

A. Mutating

B. Non-mutating

(b) (2 points) Assume this function is called on the following list: `[69, 420, 3.14159365, 474, 666]`. What is the output of this function, or the final state of the list? (Depending on your answer to part (a))

A. `[69, 420, 3.14159365, 474, 666]`

B. `[3.14159365, 69, 420, 474, 666]`

C. `[3.14159365, 69, 420, 474, 666, 69, 420, 474, 666]`

D. An `Exception` of some kind is raised

E. None of the above

6. (5 points) **Correctness**

Nugget has developed the following block of Python code:

```python
import random

def mystery():
    a = random.randint(0, 5)
    b = random.randint(0, 5) / 2
    if a < b:
        return a
    else:
        return b

print("The number is " + mystery() + ".")
```

Nugget thinks this code is correct, while UTM Victim argues the code has at least one case where it fails. Who is correct, and why?

A. Nugget is correct

B. UTM Victim is correct

Why: ____

Part II: Short Answer

Answer each question to the best of your abilities. Partial marks will be awarded for partial answers.

1. (5 points)

Part III: Long Answer

Answer each question to the best of your abilities. Partial answers $\Longrightarrow$ partial marks. Breaking any restrictions in the question will result in a mark of 0.

1. (10 points) **The Happy List**
Let L be a list. We say that L is **happy** if L is in ascending order and contains at least 2 elements which add up to an arbitrary value k. Implement the following method to determine if a list is happy. You may assume that the list is non-empty, sorted, and contains only integers.

RESTRICTIONS:

- You may **NOT** use `sets`
- Your method **MUST** be $\mathcal{O}(n)$ time (i.e. linear time)
- You may **NOT** use concepts taught outside of the scope of CSC108

```
def is_happy(L: list[int], k: int) -> bool:
    """
    Given a list, returns whether or not the list is happy.
    Precondition: L is a list of integers in ascending
    order. k is an integer.
    """
    # TODO: Implement this method
```

2. (10 points) **The Secret**
   `Ibra.Java` is hosting a secret MAT102 lecture. This lecture is so secret. that he only tells one student about it. However, his student is

Rough Work

This page will NOT be marked. You may use this page for rough work.

Rough Work

This page will NOT be marked. You may use this page for rough work.