UNIVERSITY OF IBRA
Department of Numeracy, Computation, and Probability

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

Instructors: Themba, Ibrahim

Duration: Good Luck.
Aids Allowed: God Himself.
2023/12/08

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*** *undo the mental damange it will inflict.*

This exam contains 15 pages (including this cover page) and 17 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. `NoneType`

   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

   E. An `Exception` of some kind is raised

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(lst1: list, lst2: list, call: callable) -> list:
    lst1 = [x for x in lst2[:1:-2]]
    lst2 = [x for x in lst1[1::2]]
    if lst1 == lst2:
        return lst1
    if len(lst1) > len(lst2):
        cursed_list_2 = lambda x, y: [x for x in y[:1:-2]]
    return call(lst1, lst2)

def cursed_list_2(lst1: list, lst2: list, call: callable) -> list:
    if len(lst1) >= (len(lst2)):
        cursed_list_1 = lambda x, y, z: [x for x in y[3::2]]
    return call(lst1, lst2, cursed_list_2)

print(cursed_list_2([1, 2, 3, 4, 5][::-1], [1, 2, 3, 4, 5][:2:-2],
cursed_list_1))
```

What is the output of this code?

A. `[4, 2]`

B. `[1, 3, 5]`

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

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

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

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

E. An `Exception` of some kind is raised

F. None of the above

6. (5 points) **Correctness**

Nugget has developed the following block of Python code:

```
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: _____

For full credit, if you selected "UTM Victim is correct", you must specify the ***`Exception`*** *that is raised and the error message.* _____

Part II: Short Answer

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

1. (5 points) **Object-Oriented Programming**
   Briefly explain the difference between a class and an object.

2. (5 points) **Object-Oriented Programming**
   Briefly explain what it means to be a *mutable* vs. *immutable* object.

3. (5 points) **Regex**
   Briefly explain what the following regular expression matches: `[a-zA-Z0-9]+`

4. (5 points) **Regex**
   Ibrahim is working on a new `UserContact` module for `TTBTrackr`. He needs a regex that will validate a phone number. The phone number must be in the format `xxx-xxx-xxxx`, where `x` is a digit between 0 and 9. Write a regex that will validate this phone number. Your regex should also validate the area code and should work regardless of hyphens being included.

5. (5 points) **Logic and Variables**
   Simplify the following expression as much as possible. Your expression should be logically equivalent to the original expression.
   `not (True or X and (True or x and (not y or z)) or (z and Y or (not x and not y)))`

   Note: You may recieve partial credit should you decide to show your work

6. (10 points) **Code Tracing**
Refer back to the `scooby_doo` function from **Code Tracing IV**. Explicitly state a precondition and postcondition for both `scooby_doo` and `mystery`. You may assume that `arr` is a list of integers.

7. (10 points) **Code Tracing**
Patea stores his Bitcoin private key in a file on his computer. To stop people from stealing his Bitcoin, he encrypts the file using a password. Unfortunately, on a recent Discord call he leaked his encryption function:

```
from typing import TextIO

def destroy_file(file: TextIO):
    file_contents = file.readlines()
    for i in range(len(file_contents)):
        x = file_contents[i].strip()
        new_x = ""
        for z in range(len(x)):
            new_x += chr((ord(x[z]) + z - ord('a')) % 26 + ord('a'))
        file_contents[i] = new_x

    with open("destroyed_file.txt", "w") as f:
        f.writelines(file_contents)
```

If `destroyed_file.txt` contains the following text:

```
jtuwrdubvbywhrrcjyalhrnllfrnsaqelgpv
```

What is Patea's Bitcoin private key? If this encryption is not possible to reverse, explain why.

Extra space for Q7

Part III: Long Answer

Answer each question to the best of your abilities. Partial answers $\implies$ 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). Any answer that is not $\mathcal{O}(n)$ time will receive a mark of 0.
- You may **NOT** use concepts taught outside of the scope of CSC108
- You may **NOT** create any helper methods, new objects, or new variables. You may only use the variables provided to you.

```
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
    temp1, temp2, temp3 = None, None, None
```

2. (10 points) **The Happy Numer**
Let n be a positive integer. We say that n is **happy** iff replacing n with the sum of the squares of its digits, and repeating this process, eventually leads to the number 1 within 7 iterations. For example, 19 is happy because:

$$19 \to 1^2 + 9^2 = 82$$
$$82 \to 8^2 + 2^2 = 68$$
$$68 \to 6^2 + 8^2 = 100$$
$$100 \to 1^2 + 0^2 + 0^2 = 1$$

Write an algorithm to determine whether a given number n is happy or not.

RESTRICTIONS:

- For full marks, your solution must be $\mathcal{O}(1)$ time (i.e. constant time). Any answer that is not $\mathcal{O}(1)$ time will receive a mark of 0.
- For full marks, your code must be less than 4 lines. Any answer that is more than 4 lines will receive a mark of 0.

Hint: You may want to use List Comprehensions. No, I don't care that it wasn't taught in CSC108

```
def is_happy(n: int) -> bool:
    """
    Given a number, returns whether or not the number is happy.
    Precondition: n is a positive integer.
    """
```

3. (10 points) **Malware Containment I**
`Themba.java` is working on a secret CSC108 UltraSheet™Pro Max. To help keep this textbook online, he distributes it through Peer-to-Peer (P2P) file sharing. Unfortunately, an evil TA from TMU has intercepted the file and injected malware into it. Anyone who recieves the UltraSheet from the TA will be infected. Given:

   - The initial infected user
   - A map of which user got the UltraSheet from which other user

   Write an algorithm to determine the total number of infected users.

   RESTRICTIONS:

   - For full marks, your solution must be $\mathcal{O}(n^2)$ time (i.e. quadratic time)
   - For full marks, your code must be less than 10 lines. Any answer that is more than 10 lines will receive a mark of 0.
   - You may **NOT** use concepts taught outside of the scope of CSC108

```
def num_infected(initial_infected: str,
                 user_map: dict[str, list[str]]) -> int:
    """
    Given the initial infected user and a map of which user got
the
    UltraSheet from which other user, returns the total number of
    infected users.
    Precondition: initial_infected is a string. user_map is a
    dictionary mapping strings to lists of strings.
    """
```

4. (10 points) **Malware Containment II**
   `Themba.java` has caught wind of the rogue TA infecting his UltraSheet. He has decided to implement a new way to distribute the UltraSheet. Instead of sending all the parts of the textbook to each user, he will send each user a single part of the textbook. Each user will then send their part to another user. This process will repeat until all users have all parts of the textbook. Write an algorithm to determine whether an arbitrary starting point can result in having zero malware infections.

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.

ASCII Reference Sheet

Regular ASCII Chart (character codes 0 – 127)

000d	00h	␀	(nul)	016d	10h	►	(dle)	032d	20h	␣	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	`	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah	◙	(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch		(np)	028d	1Ch	∟	(fs)	044d	2Ch	,	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh	♪	(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

Extended ASCII Chart (character codes 128 – 255) Latin1/CP1252

128d	80h	€	144d	90h		160d	A0h	␣	176d	B0h	°	192d	C0h	À	208d	D0h	Ð	224d	E0h	à	240d	F0h	ð
129d	81h		145d	91h	‘	161d	A1h	¡	177d	B1h	±	193d	C1h	Á	209d	D1h	Ñ	225d	E1h	á	241d	F1h	ñ
130d	82h	‚	146d	92h	’	162d	A2h	¢	178d	B2h	²	194d	C2h	Â	210d	D2h	Ò	226d	E2h	â	242d	F2h	ò
131d	83h	ƒ	147d	93h	“	163d	A3h	£	179d	B3h	³	195d	C3h	Ã	211d	D3h	Ó	227d	E3h	ã	243d	F3h	ó
132d	84h	„	148d	94h	”	164d	A4h	¤	180d	B4h	´	196d	C4h	Ä	212d	D4h	Ô	228d	E4h	ä	244d	F4h	ô
133d	85h	…	149d	95h	•	165d	A5h	¥	181d	B5h	µ	197d	C5h	Å	213d	D5h	Õ	229d	E5h	å	245d	F5h	õ
134d	86h	†	150d	96h	–	166d	A6h	¦	182d	B6h	¶	198d	C6h	Æ	214d	D6h	Ö	230d	E6h	æ	246d	F6h	ö
135d	87h	‡	151d	97h	—	167d	A7h	§	183d	B7h	·	199d	C7h	Ç	215d	D7h	×	231d	E7h	ç	247d	F7h	÷
136d	88h	ˆ	152d	98h	˜	168d	A8h	¨	184d	B8h	¸	200d	C8h	È	216d	D8h	Ø	232d	E8h	è	248d	F8h	ø
137d	89h	‰	153d	99h	™	169d	A9h	©	185d	B9h	¹	201d	C9h	É	217d	D9h	Ù	233d	E9h	é	249d	F9h	ù
138d	8Ah	Š	154d	9Ah	š	170d	AAh	ª	186d	BAh	º	202d	CAh	Ê	218d	DAh	Ú	234d	EAh	ê	250d	FAh	ú
139d	8Bh	‹	155d	9Bh	›	171d	ABh	«	187d	BBh	»	203d	CBh	Ë	219d	DBh	Û	235d	EBh	ë	251d	FBh	û
140d	8Ch	Œ	156d	9Ch	œ	172d	ACh	¬	188d	BCh	¼	204d	CCh	Ì	220d	DCh	Ü	236d	ECh	ì	252d	FCh	ü
141d	8Dh		157d	9Dh		173d	ADh		189d	BDh	½	205d	CDh	Í	221d	DDh	Ý	237d	EDh	í	253d	FDh	ý
142d	8Eh	Ž	158d	9Eh	ž	174d	AEh	®	190d	BEh	¾	206d	CEh	Î	222d	DEh	Þ	238d	EEh	î	254d	FEh	þ
143d	8Fh		159d	9Fh	Ÿ	175d	AFh	¯	191d	BFh	¿	207d	CFh	Ï	223d	DFh	ß	239d	EFh	ï	255d	FFh	ÿ

Hexadecimal to Binary

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special