COMP130 Final Exam, practice exam

Time allowed: *180 minutes*

Total points on exam: *140 points - allow about one minute per point, and you will have plenty of time to check your answers.*

Write solutions on blank paper, not on this exam.

No devices are permitted. No materials may be consulted, except for two pages (two sides of US letter paper) of your own handwritten notes. Solutions must employ methods we have studied in class; other approaches will receive little credit.

Question 1. Data types (8 points)

What is the data type of each of the following variables?

```
a = -9
b = str(a)
c = len(b)
d = -46.223
e = a + c
f = b + b
g = [a, b, c, d, e, f]
h = g[3]
```

Question 2. Loop output (10 points)

What is the output of the following Python code?

```
x = 3
s = 'apple'
for k in range(5):
    x = x + 2
    print(x, s)
    if x > 6:
        s = 'banana'
```

Question 3. Function calls and stack diagrams (15 points)

Consider the following Python code

```
 1  def ant(fig):
 2      leek = fig + 3
 3      return bee(leek, fig)
 4
 5  def bee(nut, yam):
 6      if nut > yam:
 7          return cow(nut - yam, nut)
 8      else:
 9          return cow(yam - nut, nut)
10
11  def cow(kale, okra):
12      plum = kale * okra
13      return plum
14
15  egg = 7
16  print(ant(egg))
```

(a) (5 points) What is the output of this code?

(b) (10 points) Draw a stack diagram for this code at the moment immediately after line 12 has been executed. (The `print` function can be omitted from the diagram.)

Question 4. Objects, methods, and constructors (12 points)

Note: For this question, you are not expected to have any familiarity with the `decimal` module. Use your generic knowledge of objects, classes, and modules to answer the questions. Consider the following code.

```
 1  import decimal
 2  from decimal import Decimal
 3
 4  my_height = Decimal('1.75')
 5  mean_height = Decimal('1.80')
 6  if my_height.compare(mean_height) > 0:
 7      print("I am taller than the mean height.")
 8  context = decimal.getcontext()
 9  my_context = decimal.Context(prec=20) # compute to 20 decimal places
10  decimal.setcontext(my_context)
11  ratio = my_height / mean_height
12  print("My height is", ratio, "times the mean height.")
13  # output: 'My height is 0.97222222222222222222 times the mean height.'
```

(a) (4 points) List all line numbers that perform a *method call* on an object.

(b) (4 points) List all line numbers that invoke a *constructor*.

(c) (4 points) List all line numbers that call a *function from a module* (excluding constructors).

Question 5. While loop and booleans (15 points)

Below is the code for a function with the signature `double_plus_one_sequence(start)`, where `start` is an integer parameter. The purpose of the function is to print out a sequence beginning with the start parameter, and each element of the sequence is calculated from the previous one by doubling the previous element then adding 1. For example, if `start` is 3, then the first few values printed are 3, 7, 15, 31,.... This is because $7 = 2 \times 3 + 1$, $15 = 2 \times 7 + 1$, and $31 = 2 \times 15 + 1$. Only values less than 100 are printed. The values are printed on a single line of output, with each value separated by a comma and a space character. Finally, the function keeps track of whether any element of the sequence is a multiple of five, and prints out a sentence stating whether or not any multiples of five were observed.

For example, when `start` is 3, the full output is:

```
3, 7, 15, 31, 63
At least one sequence element was a multiple of five.
```

And when `start` is 9, the full output is:

```
9, 19, 39, 79
No sequence elements were multiples of five.
```

Below is an implementation of `double_plus_one_sequence`. Fill in each blank in the code.

```
def double_plus_one_sequence(start):
    val = ______________
    has_multiple_of_5 = ______________
    is_first_iteration = _______________
    while ___________________:
        if _________________________:
            print(", ", end='')
        else:
            is_first_iteration = _____________
        print(val, end='')
        if ___________________:
            has_multiple_of_5 = ________________
        val = ______________________
    print()
    if has_multiple_of_5:
        print("At least one sequence element was a multiple of five.")
    else:
        print("No sequence elements were multiples of five.")
```

Question 6. Trace a recursive function (10 points)

What is printed by the following program? For full credit, explain your reasoning.

```python
def weird_function(s):
    if len(s) >= 8 or len(s) < 1:
        return s
    else:
        t = s[0] + s + s[-1]
        return weird_function(t)

print(weird_function("hello"))
```

Question 7. Dictionary manipulation (20 points)

The following code defines a dictionary storing information about the residents of an apartment building. In each key-value pair, the key is an apartment number and the value is the name of the resident living in that apartment.

```python
resident = {}
resident[512] = 'Li Wei'
resident[903] = 'Emily Thompson'
resident[262] = 'Nguyen Thi Mai'
resident[441] = 'Carlos Mendoza'
resident[681] = 'James Whitaker'
...
```

We imagine the dictionary contains a large number of key-value pairs, in addition to the ones shown above. If an apartment is unoccupied, then its key is not in the dictionary.

(a) (2 points) Assuming the dictionary contains the key `723`, write code that prints out the name of the person living in apartment 723.

(b) (2 points) Write code that changes the resident of apartment 903 to be Sofia Rivera.

(c) (2 points) James Whitaker moves out of apartment 681, leaving it unoccupied. Write code that adjusts the dictionary to represent this situation correctly.

(d) (4 points) Write some code that prints `yes` if apartment 777 is occupied and `no` otherwise.

(e) (5 points) The residents of apartments 101 and 202 swap their apartments. Write code that implements this swap.

(f) (5 points) Write some code that prints `yes` if someone called Aiko Tanaka resides in the building and `no` otherwise.

Question 8. Reorder function with loops (15 points)

This question asks you to implement a function with the signature `remove_prefixes(words)`. The parameter `words` is a list of words. The function removes from this list any elements that are a strict prefix of another element. As an example, consider the following code.

```
words = ['cat', 'dogged', 'caterpillar', 'dog', 'cater', 'cat']
remove_prefixes(words)
print(words)
```

This code prints the output `['dogged', 'caterpillar']`. The word `dog` is removed because it is a prefix of `dogged`. The words `cat` and `cater` are removed because they are prefixes of `caterpillar`.

Below is a complete implementation of the function `remove_prefixes`, with the lines in a random order and with indentation removed. Re-write the lines in the correct order and with correct indentatation.

```
for i in range(len(words)):
words_to_remove.append(words[j])
for w in words_to_remove:
if w in words:
def remove_prefixes(words):
words.remove(w)
for j in range(len(words)):
words_to_remove = []
if i != j and words[i].startswith(words[j]) and words[i] != words[j]:
```

Question 9. Social, legal, and ethical issues in computing (20 points)

(a) (5 points) Briefly describe one source of bias in artificial intelligence algorithms that was mentioned in our reading on this topic.

(b) (5 points) Define *open-source software*.

(c) (5 points) Give an example of an open-source software program that has had a significant impact on society. Briefly explain your answer.

(d) (5 points) Briefly describe one false dichotomy in computing and society that was mentioned in our reading on this topic.

The exam continues on the next page.

The final question is more challenging. Please do not attempt this question until you have completed and checked your answers to all other questions.

Question 10. Challenge: dictionary and list manipulation (15 points)

Given a Python dictionary d, let $x_1, x_2, \ldots, x_n$ be a sequence of n keys or values that are stored in d. We say $(x_1, x_2, \ldots, x_n)$ is a *chain* of d if $d[x_1] = x_2, d[x_2] = x_3, \ldots, d[x_{n-1}] = x_n$, and the x_i are all distinct.

This question asks you to write a function with the signature `longest_chain(d)`. The parameter `d` is a dictionary, and the return value should be a list containing the longest chain of `d`. If there is a tie for longest chain, any one of the longest chains may be returned.

Some examples of correct values of `longest_chain(d)`, for various dictionaries `d`, are given below.

```
d: {1: 2, 3: 4, 5: 7, 6: 11, 7: 6, 9: 10, 10: 11}
longest_chain(d): [5, 7, 6, 11]

d: {1: 2, 2: 3, 3: 4, 4: 1}
longest_chain(d): [1, 2, 3, 4]  # other answers possible here

d: {1: 2, 2: 4, 3: 1, 4: 6, 5: 6, 6: 2, 7: 5, 8: 8}
longest_chain(d): [3, 1, 2, 4, 6]

d: {3: 3}
longest_chain(d): [3]

d: {3: 3, 5: 4}
longest_chain(d): [5, 4]

d: {}
longest_chain(d): []
```

Hint: Define a helper function `chain(d,k)` which computes the chain starting at key `k` in dictionary `d`.

Total number of points: **140**