CS150 Fall 2021 – Midterm Solution

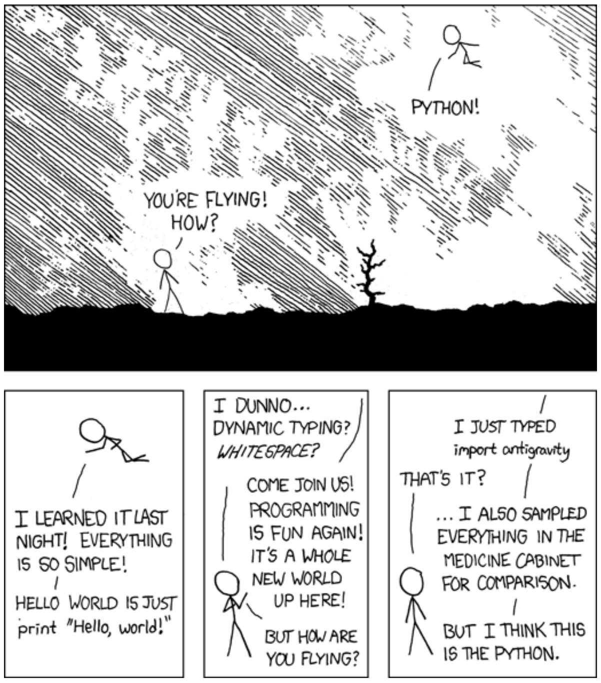
Name: ______________________________ **Section: A / B / C**

Date: ____________ **Start time:** __________ **End time:** __________

Honor Code:

Signature: ______________________

This exam is closed book, closed notes, closed computer, closed calculator, etc. You may only use (1) the midterm "cheat sheet" from the course page, and (2) a single double-sided letter sheet of notes of your own creation. **You have 2.5 hours.** Read the problem descriptions carefully and write your answers clearly and *legibly* in the space provided. Circle or otherwise indicate your answer if it might not be easily identified. You may use extra sheets of paper, stapled to your exam, if you need more room, as long as the problem number is clearly labeled and your name is on the paper. If you attached extra sheets indicate on your main exam paper to look for the extra sheets for that problem.
You do need to include module imports (if relevant for your code), but do not need to include comments, docstrings or constants in your code.



PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Question	Points	Score
Warming up	19	
Slice and dice	12	
Function calls	8	
T/F	8	
We've got problems	16	
Coding	16	
Turtle fun	10	
Total:	89	

Question 1: Warming up [19 points]

(a) (3 points) Which of following instructions in a recipe show why recipes can be an imperfect analogy for algorithms in a Computer Science context.

√ **Add salt to taste**
◯ Bake at 375°F for 30 minutes
◯ Add 1 teaspoon of granulated white sugar
◯ Add 400 grams of all-purpose flour

Briefly explain your choice.

Solution: In contrast to the other answers, "Add salt to taste" is ambiguous and open to interpretation. Every step in an algorithm needs to be precisely specified.

(b) (3 points) You wrote a function that takes a list of integers as a parameter and requires the list is non-empty (i.e., has at least one element). Which of the following would you use to document that requirement?

√ **The docstring**
◯ A block/inline comment (starting with the # symbol)

Briefly explain your choice.

Solution: Since anyone using the function would need to know that requirement (so they can provide valid arguments), you should describe that limitation in the docstring. Any block/inline comments would only be visible to someone reading the code implementing your function.

(c) (5 points) You are doing a set of experiments on biased dice. Write a function named `dice` that simulates a biased 6-sided die (with the numbers 1-6) that rolls a 6 with twice the probability of the other numbers, i.e., the probability of rolling a 6 is 2/7 and the probability of rolling all other numbers is 1/7. `dice` should have no parameters and return an integer in the range [1,6] (1-6 inclusive). Here are some sample calls and output.

```
>>> dice()
2
>>> dice()
6
```

Solution:

```
from random import randint

def dice():
    roll=randint(1,7)
    if roll == 7:
        return 6
    else:
        return roll
```

(d) (8 points) Quick coding: Write two functions, one using a `for` loop and the other using a `while` loop, to print the numbers between 1 to 99 inclusive, in ascending order, one number per line, that are multiples of 11. Your functions should take no parameters and return no values. For full credit your functions should be as concise and efficient as possible.

Solution:

```
def forloop():
    for i in range(11, 100, 11):
        print(i)
```

```
def whileloop():
    i = 11
    while i <= 99:
        print(i)
        i += 11
```

Question 2: Slice and dice [12 points]

Given the variables `s` and `t` with the following values:

```
>>> s
'strange year'
>>> t
'at college'
```

Evaluate the following expressions and provide the resulting value in the boxes, one character per box. Shade in any unused boxes at the end of the string. Make sure upper case letters can be clearly distinguished from lower case letters.

(a) `t[3]+s[0]+t[2]+s[:3]`

Solution: `cs str`

(b) `s[::3] + t[::4]`

Solution: `saeeaog`

(c) `s.upper()[-4:-1]+t[-8:]`

Solution: `YEA college`

(d) `(s[2:4] + "h")*2 + t[1:1]`

Solution: `rahrah`

Question 3: Function calls [8 points]

Consider the following Python code:

```
def bar(x, n):
    print(x * n)
    return int(x) * n

def foo(s):
    r = 0
    for i in range(len(s)):
        r += bar(s[i], i+1)
    return r

y = foo("8501")
```

(a) After execution the value of `y` is:

(a) **22**

(b) What if anything is printed during execution?

Solution:

```
8
55
000
1111
```

Question 4: T/F [8 points]

For each of the statements below state whether they are **T** (true) or **F** (false).

(a) __**T**__ The expression `if b == True` can be simplified to `if b`.

(b) __**T**__ It is possible that a given while loop does not execute any iterations

(c) __**F**__ `(7 / 4) * 4 + 7 % 4` evaluates to 7.0

(d) __**F**__ The following function will return the maximum value in *any* valid non-empty list of integers provided as the argument

```
def max(sequence):
    cur_max = -1
    for val in sequence:
        if val > cur_max:
            cur_max = val
    return cur_max
```

(e) __**F**__ After this code executes `x` has the value 4

```
x = 0
for i in range(4):
    for j in "a"*i:
        x += 1
```

(f) __**T**__ The following code will execute without an error for *any* list or string argument

```
def mystery(arg):
    for i in range(len(arg)):
        print(arg * i)
```

(g) __**F**__ The following loop will eventually terminate for *any* input provided by the user

```
i = input("Enter your name: ")
while len(i) >= 0:
    i = i[1:]
```

(h) __**T**__ For the following value of `a`, `a[2][0] == a[0][2]` evaluates to `True`

```
a = [[1, 2, 3], [2, 4, 5], [3, 5, 6]]
```

Question 5: We've got problems [16 points]

(a) The function below has two integer parameters, `a` and `b`. The function works as desired, however, it uses bad coding style. Rewrite the function to have identical behavior (i.e., for all possible values of `a` and `b` return the same value), but to be as concise as possible and implemented with good style.

```
def could_be_better(a, b):
    if 8 <= b <= -8:
        return False
    elif a < 10 and b < 10:
        return True
    elif a <= 9 and b <= 9:
        return False
    elif a >= 10 or b >= 10:
        return False
    else:
        return True
```

Solution:

```
def better(a, b):
    return a < 10 and b < 10
```

(b) The following function was designed to return an easy-to-read string with comma separators every 3 digits for its integer parameters, e.g. return "12,345" for 12345 and "123" for 123. There are 3 problems with this code. Identify and briefly describe (referencing the line numbers) i) one syntax error, ii) one runtime error (syntactically valid Python that generates an error when actually executed) and iii) one logical error (the code would execute to completion if the other errors are fixed but produces incorrect results), for three errors total. The errors should not be variations of the same issue and should impact correctness, not just style. You do not need to fix the errors.

```
1  def pretty_int(num):
2      digits = str(num)
3      separated_digits = 0
4      while len(digits) greater 3:
5          separated_digits = "," + digits[-3:] + separated_digits
6          digits = digits[:-3]
7      return separated_digits + digits
```

i. Syntax Error:

Solution: On line 4 `greater` is not a valid operator in Python (even though `and` and `or` are), it should be `>`.

ii. Runtime Error:

Solution: On line 3 `separated_digits` is initialized as integer, but we subsequently attempt to concatenate it with a string inside the loop on line 5 and on line 7 after the loop. It should have been initialized with the empty string.

iii. Logical Error:

Solution: On line 7 in the final return statement we append `digits` to `separated_digits` when we should prepend, i.e., as written the the function returns ",23412" for 12345 instead "12,345". It should be `return digits + separated_digits`.

Question 6: Coding [16 points]

As part of a study of deer habitat you fitted a deer with a tracker that records its position to a file every 10 seconds. Each position is recorded as integer x, y coordinates in meters. The x and y coordinates are separated by a space, like shown in the example file snippet below.

```
100 70
120 100
60 -50
```

Write a function named `distance` that takes one parameter, the filename, and returns the approximate distance the deer traveled. For simplicity you will calculate the total distance traveled as the sum of the straight-line distances between all of the position readings. Recall that the distance between two points is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. For the example file above, you would calculate the total distance as the sum of the distances from $(100, 70)$ to $(120, 100)$ and from $(120, 100)$ to $(60, -50)$. Your implementation can include other functions if that is helpful to you but doing so is not required. You can assume the file always has at least two lines. As a hint, recall that the string 'split' method can generate a list of words from a string, e.g., `"100 100".split()` returns `["100","100"]`.

Solution:

```
from math import sqrt

def distance(filename):
    x = []
    y = []
    with open(filename, "r") as file:
        for line in file:
            coords = line.split()
            x.append(int(coords[0]))
            y.append(int(coords[1]))

    total_distance = 0
    for i in range(1, len(x)):
        total_distance += sqrt((x[i] - x[i-1])**2 + (y[i] - y[i-1])**2)
    return total_distance
```

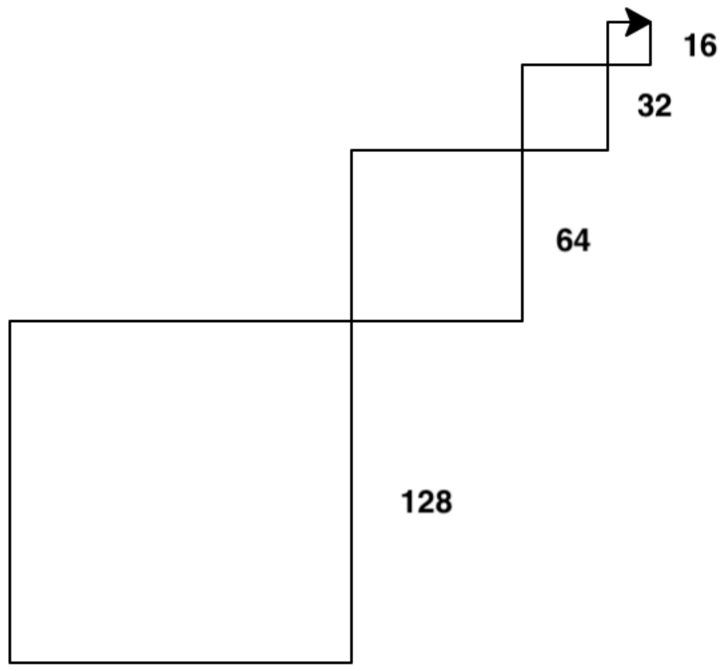
Question 7: Turtle fun [10 points]

```
from turtle import *

def shape(x):
    for i in range(6):
        forward(x)
        left(90)

side = 128
while side > 10:
    shape(side)
    right(180)
    side = side // 2
```

Using the grid below, draw the image that would be created by the above code. Label your grid so all relevant dimensions and angles are clear. Assume that the turtle is initially at the lower left corner, facing right. Assume each grid square is 16 pixels by 16 pixels.



Page intentionally left blank.