CSC 110

Midterm Exam:
Thursday, 4 November 2021

Name:_____(please print clearly!)

UVic ID number:_____(please print clearly!)

Signature:_____

Exam duration: 50 minutes

Instructor: Celina Berg

Students must check the number of pages in this examination paper before beginning to write, and report any discrepancy immediately.

- We will not answer questions during the exam. If you feel there is an error or ambiguity, write your assumption and answer the question based on that assumption.
- Answer all questions on this exam paper.
- The exam is closed book. No books or notes are permitted.
- **Electronic devices, including calculators, are not permitted**.
- The marks assigned to each part are printed within brackets. Partial marks are available.
- There are eight (8) pages in this document, including this cover page.
- Page 6 is left blank for scratch work. If you write an answer on that page, clearly indicate this for the grader under the corresponding question.
- Clearly indicate only one answer to be graded. Questions with more than one answer will be given a **zero grade**.
- It is strongly recommended that you read the entire exam through from beginning to end before beginning to answer the questions.

Part 1 (13 marks)
For the following questions, write your final answer in the box provided.
Write **"invalid"** for those with syntax errors or with infinite loops.

a) What are the values of the variables a and b after the following code segment has executed?
(1 mark each box)

```
a = 0
b = 5

while a<8 and b>0:
    a = a + b
    b = b – 2
```

a: 8

b: 1

b) What are the values of the variables a and b after the following code segment has executed?
(1 mark each box)

```
a = 0
b = 5

while a<8 or b>0:
    a = a + b
    b = b – 2
```

a: 9

b: -1

c) Given the following function definition (documentation omitted intentionally):

```
def foo(lon: list[int]) -> int:
    count = 0
    for index in range(len(lon)):
        if array[index] % 2 != 0:
            array[index] *= 2
            count += 1

    return count
```

What is the output of the following code segment:

```
the_list = [2, 3, 1, 6]
count = 4
foo(the_list)

print(the_list[0])
print(the_list[3])
print(count)
```

(2 marks for answer "invalid" because array should be lon within foo)

2
6
4
(1 mark if answer above)

d) Given the following function definition (documentation omitted intentionally):

```
def foo(lon: list[int]) -> None:
    index = 0
    for val in lon:
        if val % 2 == 0:
            lon[index] *= 2
```

What is the output of the following code segment:

```
the_list = [2, 3, 1, 6]
foo(the_list)
print(the_list)
```

8 3 1 6 (correct – 3 marks)
4 3 1 12 (incorrect – 2 marks)

e) The function definition below is missing some documentation. Trace the following valid function call to help you understand the function behaviour:

```
mystery([1, 2, 3], [5, 7, 2])
```

Complete the missing documentation for the function including:

- return type hint
- purpose
- examples demonstrating full test coverage

```
def mystery(list1: list[int], list2: list[int]) -> list[int]:
    """
    Creates a list combining alternating values from list1 and list2
    until either list1 or list2 do not have any values left.
    Return the populated list.
    >>> mystery([], [])
    []
    >>> mystery([2, 3, 5], [])
    []
    >>> mystery([], [4, 5, 6])
    []
    >>> mystery([2, 3, 8], [4, 5, 6])
    [2, 4, 3, 5, 8, 7, 6]
    """
    result = []
    n1 = len(list1)
    n2 = len(list2)
    n  = min(n1, n2)

    for i in range(n):
        result.append(list1[i])
        result.append(list2[i])

    return result
```

Rubric:

/1 mark return type hint
/1 purpose – must reasonably describe that values are being joined into a single result list with values from the input lists.
/2 marks – test cases
0.5 marks each (test and expected must be correct):
- 2 empty lists
- list1 empty, list2 non-empty
- list1 non-empty, list2 empty
- list1 non-empty, list2 non-empty

Part 2 (15 marks)

For the following two questions on the next two pages you will use the following type alias:

```
# represents a Car as (make, model, maximum speed, customer rating)
# where make!= '', model!='', speed>0 in km/hr and rating is in range [1,5]
Car = tuple[str, str, int, int]
MAKE   = 0
MODEL  = 1
SPEED  = 2
RATING = 3
```

a) (5 marks) Consider the following function definition with **BLANKs** to indicate missing code:

```
def find_car(locars: list[Car], rating: int, speed: int) -> int:
    """ Returns the index of the first car in locars that has
    a customer rating at least as high as the given rating
    and a maximum speed at least as high as the given speed.
    The function should return -1 if no Cars in locars have
    the necessary rating and speed

    >>> cars= [('Honda', 'Prelude', 160, 3), ('Volkswagen', 'Jetta', 150, 5),
    ('Toyota', 'Highlander', 180, 5), ('Mercedes', 'C300', 200, 4)]
    >>> find_car([], 4, 160)
    -1
    >>> find_car(cars, 4, 150)
    1
    >>> find_car(cars, 4, 181)
    3
    """
    index = 0
    num_elements = len(locars)

    while ( BLANK 1 ):
        index += 1

    if ( BLANK 2 ):
        index = -1

    return index
```

In the boxes below, write the code that should replace **BLANK 1** and **BLANK 2** respectively.
NOTE: marks will not be given for re-writing the function or adding other code to the function.

BLANK 1:

1 mark – index<num_elements (must be first combined with an ***and***)
2 marks – comparison of locars elements to rating and speed
-1 if missing [index] or nested index [index[RATING]],
-0.5 wrong comparator, did not use constants, off by one
1 mark – ***or*** or ***not(… and …)*** to combine conditions (must be wrapped in brackets)

Two possible correct answers:

```
(index<num_elements
and (locars[index][RATING]<rating or locars[index][SPEED]<speed))
(index<num_elements
and not(locars[index][RATING]>=rating and locars[index][SPEED]>=speed))
```

BLANK 2:

1 mark - either one of the following (½ if = instead of ==, or if comparison is off by one):

```
index == num_elements
index >= num_elements
```

b) (10 marks) Complete the missing **type hints** and **function body** according to documentation. You **must not** use Python's list comprehensions, list or string methods or any functions listed in `dir(__builtins__)` with the exception of the `len` and `append`. A zero grade for these solutions will be given. You are free to design your own helper functions.

```
def get_above_avg_cars(loc: list[Car]) -> list[str]:
    """ creates and returns a new list of the model names of only the Cars from loc
    with a customer rating that is above the average rating of all Cars in loc
    >>> cars = [('Honda', 'Prelude', 160, 3), ('Volkswagen', 'Jetta', 150, 5),
    ('Toyota', 'Highlander', 180, 5), ('Mercedes', 'C300', 200, 4)]
    >>> get_above_avg_cars([])
    []
    >>> get_above_avg_cars(cars)
    ['Jetta', 'Highlander']
    """
    # model solution, other solutions possible
    cars = []
    if locars == []:
        return cars

    avg_rating = get_average_rating(locars)
    for car in locars:
        if car[RATING] > avg_rating:
            cars.append(car[MODEL])
    return cars

def get_average_rating(locars: list[Car]) -> float:
    """ return average rating of cars in locars
    Precondition: locars not empty
    >>> get_average_rating([('Toyota', 'Highlander', 180, 5)])
    5.0
    >>> get_average_rating([('Honda', 'Prelude', 160, 3), \
    ('Volkswagen', 'Jetta', 150, 5), ('Toyota', 'Highlander', 180, 5), \
    ('Mercedes', 'C300', 200, 4)])
    4.25
    """
    total = 0
    for car in locars:
        total += car[RATING]
    return total/len(locars)
```

rubric

```
/2 marks type hints (1 per type hint)
  -0.5 if Cars, not Car
  -1 if no type specified within the []s
/3 marks calculate average
  -1 division by zero with empty list input
  -1 division within the loop instead of after the loop
/1 marks initialize and return a result list
/1 marks loops the correct number of times
  -0.5 if miss increment within while loop
  -1 if refer to element as an index in for loop header
/3 marks conditionally append to the result list
   -2 missing the condition
   -1 error in condition
   -1 append the wrong portion of the tuple
  -2 incorrect variable access when attempting to access tuple value
  -0.5 if does not use constants
   -1 if calculating average every time the loop iterates
```

Part 3 (5 marks)

The following function has errors in it.
Find and fix the errors by crossing out incorrect code and writing the code that would replace it.
NOTE: there are no errors in the documentation

```
def get_number(limit: int) -> int:
    """
    Repeatedly asks the user to enter a positive whole number
    smaller than the given limit and returns the valid input as an integer.

    Examples omitted intentionally – you should not add them.
    """

    prompt = 'Enter a positive integer smaller than: ' + str(limit)


    n = input(prompt)

    ~~while int(n) >= limit and not(n.isdigit()):~~
    while not(n.isdigit()) or int(n) >= limit:

        n = input(prompt)


        ~~return n~~
    return int(n)
```

```
rubric
/1 mark – converts limit to str
/1 mark – replace and with or
/1 mark – reverse order of expressions in while so not(n.isdigit()) behaves as guard
/1 mark – updates n within while loop
/1 mark – un-indents return + converts n to int (½ mark each)

-0.5 addition of unnecessary code or code the generates an error
  - adding condition (int(n) < 0)
  - unnecessary prompt inside the loop
  - adding return statement before gathering input in the loop
  - change comparator from >= to >
```

END OF EXAM

For grading purposes, do not fill in:

Part	Value	Mark
1	13	
2a	5	
2b	10	
3	5	
Total	**33**	