repl

EXAMINATION INSTRUCTIONS

This examination has 17 pages. Check that you have a complete paper.
You have 3 hours to complete this examination. Use your time accordingly.

READ AND OBSERVE THE FOLLOWING RULES:

- Print your name and UB Person Number, and sign your name, on the next page.
- Show all of your work in arriving at an answer, unless instructed otherwise. Partial credit will be awarded as appropriate.
- Put your answers in red text.
- Do not remove any pages or mutilate this booklet in any way.
- CAUTION – Candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.
  - Communicating, by any means, with anyone other than the course staff about the exam or course content during the exam time (even if one party has submitted the exam).
  - Making use of any resources, including but not limited to notes, calculators, computers, and cell phones.

---------- DO NOT WRITE BELOW THIS LINE! ----------

	Q1	Q2	Q3	Q4	Subtotal
MODULE 1	/28	/28	/28	/28	/112
	Q1	Q2	Q3	Q4	Subtotal
MODULE 2	/26	/26	/26	/26	/104
	Q1	Q2	Q3	Q4	Subtotal
MODULE 3	/18	/18	/18	/18	/72
	Q1	Q2	Q3	Q4	Subtotal
MODULE 4	/8	/8	/8	/8	/32

Module 1 - Question 1 [28 points, 7 points each]

The code given below is syntactically correct.

For each of the following give one ***and only one*** example of each from the code below. For each item, write out the corresponding code in the space provided. If you believe no example exists, write "*no example*" next to that item in the list. To show you how I want the question answered, *the first one is done for you.*

0. an expression with a Boolean value

```
d[k] < high
```

1. a parameter list

(lst, k, low, high)

2. an argument in a function call

d

3. assignment statement

acc = []

4. a conditional statement *(entire statement)*

```
if d[k] >= low and d[k] < high:
    acc.append(d)
```

5. a keyword

def

```
def filterRange(lst, k, low, high):
    acc = []
    for d in lst:
        if d[k] >= low and d[k] < high:
            acc.append(d)
    return acc
```

Module 1 - Question 2 [28 points total, 14 each part]

Study the following code, then answer the question which follows.

```
def mystery(a, b, c, d):
    ans = []
    if a < b and c > a + d:
        ans = [b, d]
    if a < c:
        ans.append(c)
    else:
        ans.append(d)
    return ans
```

Part A [14 points] [ ] 14 points: correct [ ] 3 points: wrong but plausible [ ] 0 points: otherwise

What does the following statement print?

```
print(mystery(1,2,7,4))
```

Write your answer below:

[2,4,7]

Part B [14 points] [ ] 14 points: correct [ ] 3 points: wrong but plausible [ ] 0 points: otherwise

What does the following statement print?

```
print(mystery(1,2,-7,-14))
```

Write your answer below:

[2,-14,-14]

Module 1 - Question 3 [28 points total]

Here's a description of how the price of a pizza at a mythical pizza place is calculated:

Your pizza price is calculated as follows:

- ***size*** *– The size is "S", "M" or "L", with base prices of $8, $12, and $16, respectively*
- ***toppings*** *– each topping costs one quarter the base price of the pizza.*
- ***extra cheese*** *– True or False: extra cheese costs half the base price of the pizza.*

Consider a function named 'pizzaPrice' which will be called with three arguments, a string ("S", "M" or "L") representing the size, an integer representing the number of toppings, and a Boolean determining whether the piazza has extra cheese or not. This function computes the price for the pizza using the above rules.

Describe briefly what a test case is: [14 points total]
[ ] 14points: perfect [ ] 7 points: clearly wrong, some correct elements
[ ] 10 points: essentially correct but with small mistakes [ ] 0 points: for anything else

A test case is a set of inputs to a function or piece of code, along with the expected outcome of executing the code with those inputs.

Give a test case for the pizzaPrice function: [14 points total]
[ ] 14points: perfect [ ] 7 points: clearly wrong, some correct elements
[ ] 10 points: essentially correct but with small mistakes [ ] 0 points: for anything else

pizzaPrice("S", 1, True) should return 14

Module 1 - Question 4 [28 points total]

[ ] 28 points: perfect
[ ] 20 points: essentially correct, small mistakes
[ ] 10 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Define the pizzaPrice function from the previous question. Write your answer below:

```
def pizzaPrice(s, t, c):
  if s == "S":
    base = 8
  elif s == "M":
    base = 12
  else:
    base = 16
  total = base + base * 0.25 * t
  if c:
    total = total + base * 0.5
  return total
```

Module 2 - Question 1 [26 points, 5.2 points each]

The code given below is correct: it compiles without errors.

For each of the following give one ***and only one*** example of each from the code below. For each item, write out the corresponding code in the space provided. If you believe no example exists, write "*no example*" next to that item in the list. To show you how I want the question answered, *the first one is done for you.*

0. string literal

```
no example
```

1. conditional statement (entire statement)

```
if (prevItem == undefined) {
        prevItem = item;
} else {
        answer.push(item);
        answer.push(prevItem);
        prevItem = undefined;
}
```

2. looping statement (entire statement)

```
for (let item of anArray) {
        if (prevItem == undefined) {
                prevItem = item;
        } else {
                answer.push(item);
                answer.push(prevItem);
                prevItem = undefined;
        }
}
```

3. an assignment statement

```
prevItem = item
```

4. an argument list

```
(item)
```

5. an expression with a Boolean value

```
prevItem == undefined
```

```
function mystery(anArray) {
    let answer = [];
    let prevItem = undefined;
    for (let item of anArray) {
        if (prevItem == undefined) {
            prevItem = item;
        }
        else {
            answer.push(item);
            answer.push(prevItem);
            prevItem = undefined;
        }
    }
    return answer;
}
```

Module 2 - Question 2 [26 points total]

Consider a Python function named `keysWithValue` which has two parameters. The function will be called with a dictionary and a value. The function will return a list of those keys which are paired with the indicated value. You may assume NOT that the value is definitely present in the dictionary.

For parts 1, 2, and 3, assume the following definitions have been made:

```
myDict = { 'a':17, 'b':24, 'c':17, 'd':17, 'e':90, 'f':90 }
```

Part 1 [2 points]
What value does `numberWithKeyValue(myDict, 17)` return?

['a', 'c', 'd']

Part 2 [2 points]
What value does `numberWithKeyValue(myDict, 24)` return?

['b']

Part 3 [2 point]
What value does `numberWithKeyValue(myDict, 38)` return?

[]

Part 4 [20 points]
[ ] 20 points: perfect
[ ] 14 points: essentially correct but with small mistakes
[ ] 7 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Define, in Python, the function `keysWithValue`:

```
def keysWithValue(d, v):
  result = []
  for k in d.keys():
    if d[k] == v:
      result.append[k]
  return result
```

Module 2 - Question 3 [26 points total]

[ ] 26 points: perfect
[ ] 18 points: essentially correct, small mistakes
[ ] 9 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Define a JavaScript function named `merge` with two parameters. Assume the function will be called with two lists of equal length. The lists will contain string values. Define the function so it returns an object whose key-value pairs are constructed from the values in the two argument lists; the first list contains the keys, which will be paired with the corresponding entry from the second list.

For example,

```
merge([],[])
```

must return

```
{ }
```

whereas

```
merge(['a', 'b'],[ 'x', 'y'])
```

must return

```
{ 'a' : 'x', 'b' : 'y' }
```

Write your answer below:

```
function merge(k, v) {
  let result = {};

  for (let i = 0; i < k.length; i = i + 1) {
    result[k[i]] = v[i];
  }
  return result;
}
```

Module 2 - Question 4 [26 points total]

[ ] 26 points: perfect
[ ] 18 points: essentially correct, small mistakes
[ ] 9 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Study the following code:

```
import csv

def mystery(filename):
    with open(filename, "r", newline='') as f:
        reader = csv.reader(f)
        for line in reader:
            print(line[1]+":"+line[3]+":"+line[0])

mystery("f.csv")
```

Give possible contents for the file f.csv which would cause the above to print:

```
heart:12:always
liver:35:often
spleen:17:never
```

Give your answer below:

always,heart,???,12

often,liver,???,35

never,spleen,???,17

Module 3 - Question 1 [18 points, 4.5 points each]

For each item labelled (a) though (e), match it with its *best* description from items (1) through (9). Note that four of the items from the numbered list will not be used.

(a) script tag
(b) JSON
(c) salt
(d) AJAX get request

(1) a way to uniquely identify an HTML element <- id
(2) a way for a client to request data from a server
(3) a function to process a response from an AJAX request <- callback
(4) used to guarantee uniqueness of password hashing
(5) a way for a client to send data to a server <- AJAX post request
(6) a way for a client to obtain code from a server
(7) a way to encode data values as strings
(8) a function to translate to and from JSON <- JSON.parse, JSON.stringify, json.load, json.dump
(9) a way to encrypt data transmissions

Give you matches below:

(a) best matches with ___6___

(b) best matches with ___7___

(c) best matches with ___N/A (but the answer would be 4)___

(d) best matches with ___2___

Module 3 - Question 2 [18 points total]

[ ] 18 points: perfect
[ ] 12 points: essentially correct, small mistakes
[ ] 6 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Define a Python function named `sumData`. This function must have the annotations needed so that it handles an AJAX POST request along the route `'/sum'`. Assume that the data sent in the POST request is a list of numbers. Define `sumData` so that computes and returns the sum of the values in the list, encoded in JSON.

To recover the JSON representing the data sent as part of the POST request, use:

```
jsonBlob = bottle.request.body.read().decode()
```

For example, if `jsonBlob` is

```
'[ 12 , 24 , 4 ]'
```

then the function must return

```
'40'
```

Write your answer below:

```
import bottle
import json


bottle.post("/sum")

def sumData():

  jsonBlob = bottle.request.body.read().decode()

  lst = json.loads(jsonBlob)

  sum = 0

  for i in lst:

    sum = sum + i

  return json.dumps(lst)
```

Module 3 - Question 3 [18 points total]

[ ] 18 points: perfect
[ ] 12 points: essentially correct, small mistakes
[ ] 6 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Define a JavaScript function named `dictionary2string` with one parameter, a dictionary. This function must return a string representing the contents of the dictionary {k1:v1, k2:v2, … kN:vN}, as demonstrated by the following examples:

`dictionary2string({})` must return the string `"{}"`
`dictionary2string({"a":"b"})` must return the string `"{\n\ta->b\n}"`
`dictionary2string({"a":"b", "c":"d"})` must return the string `"{\n\ta->b\n\tc->d\n}"`

Note that `\n` denotes the newline character, and `\t` denotes the tab character; to clarify, `console.log(dictionary2string({"a":"b", "c":"d"}))` will print:

```
{
	a->b
	c->d
}
```

Write your answer below:

```
function dictionary2string(dict) {
  let s = "{"

  for (let k of Object.keys(dict)) {
    s = s + "\n\t" + k + "->" + dict[k];
  }
  if (Object.keys(dict).length > 0) {
    s = s + "\n";
  }
  s = s + "}"
  return s;
}
```

Module 3 - Question 4 [18 points total]

[ ] 18 points: perfect
[ ] 12 points: essentially correct, small mistakes
[ ] 6 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Explain in your own words what JSON is, and how it is used in transmitting/receiving data across a network connection.

JSON is a format for converting data types in our programs into strings. These strings can then be sent across a network connection and received / translated by a different program. All programs can understand strings.

Module 4 - Question 1 [8 points total]

[ ] 8 points: perfect
[ ] 6 points: essentially correct, small mistakes
[ ] 4 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

You are working for a library. Patrons can borrow books for two weeks at a time, with a limit of ten books at a time.

The library's software stores which books each of their patrons has borrowed. The software maintains an array of patron records. Each patron record is an object with keys `'patron'` and `'books'`. The value paired with `'books'` is an array of the books the patron currently has checked out.

In JavaScript, define a comparator function which orders patron records by the number of books they have signed out, from greatest to least; for patron records with the same number of books checked out, order by the title of the first book, alphabetically.

Not something we talked about in detail…but:

```
function comp(a, b) {
  if (a[“books”].length != b[“books”].length) {
    return b[“books”].length – a[“books”].length;
  } else if (a[“books”][0] < b[“books”][0]) {
    return -1;
  } else {
    return 1;
  }
}
```

Module 4 - Question 2 [8 points total]

Part A [4 points]

[ ] 4 points: perfect
[ ] 3 points: essentially correct, small mistakes
[ ] 2 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Give a concrete example of something a user could type as input to a Web application that could potentially lead to HTML injection.

 hello

Part B [4 points]

[ ] 4 points: perfect
[ ] 3 points: essentially correct, small mistakes
[ ] 2 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

What is the root cause of both HTML and SQL injection attacks?

The root cause is allowing users to provide input, but not sanitizing that input.

Module 4 - Question 3 [8 points total]

Part A [4 points]
[ ] 3 points: perfect
[ ] 2 points: essentially correct, small mistakes
[ ] 1 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Explain briefly how public key encryption works.

N/A

Part B [4 points]
[ ] 3 points: perfect
[ ] 2 points: essentially correct, small mistakes
[ ] 1 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Explain what SQL injection is, what enables the attack, and how it can be prevented?

SQL injection is when users can inject SQL commands/code into an application that uses an SQL database, such that their input modifies the database in unintended ways. It can be prevented by making sure their input is not interpreted as SQL.

Module 4 - Question 4 [8 points total]

[ ] 8 points: perfect
[ ] 6 points: essentially correct, small mistakes
[ ] 4 points: clearly wrong, some correct elements
[ ] 0 points: for anything else

Explain in your own words *why* merge sort is a more efficient sorting algorithm than selection sort.

If you merge sort a list of size n, it will require n log n steps. For selection sort on the same list, it would require n^2 steps. n^2 is larger than n log n, and it grows at a faster rate if we were to increase the size of our list.

PRINT YOUR NAME AND PERSON NUMBER:

Last Name: _____

First Name: _____

Person number: _____

SIGNATURE: _____