# Introduction to Computer Programming

| | |
|---|---|
| **First Name** | |
| **Last Name** | |
| **NetID** | |
| **Section** | |

## Instructions

- **Duration**: 90 minutes
- **Max score**: 100 pts
- Computers, calculators, phones, textbooks or notebooks are **not allowed**

- **Show your NYU ID card** to the proctor before signing the attendance sheet.
- **Turn off and place your mobile devices** with your belongings at the front of the room.
- Please **do not damage** the exam paper.
- **Do not discuss** this quiz with anyone until the test is handed back.
- If you need to ask questions, please **raise your hand and wait** for the instructor or proctor to come over.
- If you finish your exam early, please **bring your exam quietly** to the front of the room, otherwise **stay seated** until the proctor has collected all exams.
- If you brought one, **submit your cheat sheet** with your exam paper.
- The last two pages are blank and can be used as **scratch paper**, but must be turned in. The other pages must remain stapled.

Please sign the statement below:

I, ________________, confirm that I am a student in this class and I will complete this exam without accessing any unauthorized information during the exam. I have read and understood the exam instructions above.

## Question 1 | Types and Values

Consider the following four (4) variables and their values.

```
point = 10.0
word = "hello"
number = 2
sentence = "1_12_123"
```

In the table below, fill in the type and the value of each variable after being assigned the expression. The first one is done for you.

| Variable assignment | Type | Value |
|---|---|---|
| `a = word` | `string` | `'hello'` |
| `b = number * point` | | |
| `c = int(point) / number` | | |
| `d = int(point) // number` | | |
| `e = - number ** number` | | |
| `f = int(point) % 2` | | |
| `g = point // number + point % number` | | |
| `h = len(word + str(number)) + number` | | |
| `i = sentence[number:5]` | | |
| `j = str(number) + str(point)` | | |
| `k = sentence[-3:]` | | |
| `l = sentence[::2]` | | |
| `m = word * number` | | |
| `n = sentence > str(number)` | | |
| `o = sentence[-3:3]` | | |
| `p = chr(ord(word[1]) + number)` | | |
| `q = sentence.strip("1")` | | |
| `r = word < word[1:] or word < word[-2:]` | | |
| `s = sentence.split('_')[1][1]` | | |

## Question 2 | Predict Output

Predict the printed output obtained when running the following programs.

| Program 1 | Output 1 |
|---|---|
| `L1 = [1, 2, 3, 4]`<br>`print(L1[1:][1])` | |
| **Program 2** | **Output 2** |
| `L2 = ['X', 'Y']`<br>`L2.append(['XY'])`<br>`print(L2)` | |
| **Program 3** | **Output 3** |
| `L3 = ['X', 'Y']`<br>`L3.extend('AB')`<br>`print(L3)` | |
| **Program 4** | **Output 4** |
| `L4 = [10, 20, 30, 40]`<br>`for i in range(len(L4)):`<br>`    L4.append(i)`<br>`print(L4)` | |
| **Program 5** | **Output 5** |
| `D5 = {'burger': 3, 'salad': 7}`<br>`D5['burger'] = D5['salad']`<br>`D5['salad'] = 1`<br>`print(D5)` | |
| **Program 6** | **Output 6** |
| `L6 = [1, 2, 3, 4, 5]`<br>`DS = {x: x**2 for x in L6 if x > 2}`<br>`print(DS)` | |
| **Program 7** | **Output 7** |
| `D7 = {}`<br>`for v in range(12):`<br>`    D7[v//3] = v`<br>`print(D7)` | |
| **Program 8** | **Output 8** |
| `L8 = [1, 2, 3, 4, 5, 4, 3, 2, 1]`<br>`S8 = {x**2 for x in L8 if ((x % 2) != 0)}`<br>`print(S8)` | |
| **Program 9** | **Output 9** |
| `S9 = set([1, 2, 3, 4, 4, 2])`<br>`S9.add(5)`<br>`S9.remove(3)`<br>`print(S9)` | |

## Question 3 | Mystery Function

Consider the following program:

```
def mystery(d):
    d2 = {}
    for k in d.keys():
        l = []
        v = d[k]
        while v not in l:
            l += [v]
            v = d.get(v, v)
        l.sort()
        d2[k] = l
    return d2
```

What is the printed output for the following input values?

| | |
|---|---|
| **3.a** | `print(mystery({11: 1}))` |
| | |
| **3.b** | `print(mystery({1: 8, 3: 1, 11: 10}))` |
| | |
| **3.c** | `print(mystery({1: 3, 12: 5, 2: 12, 5: 1}))` |
| | |
| **3.d** | `print(mystery({9: 2, 5: 8, 3: 6, 7: 6, 6: 3}))` |
| | |

## Question 4 | Web log

A web server log tracks the IP addresses visited by users using the NYU network. The log file has the format shown below where each line contains three items separated by a semicolon. In order these items are:

- a netid followed by a colon (':'),
- the IP address of a website visited by the person with that netid, followed by a dash ('-'), and
- the timestamp consisting of the date and time of the visit (in this problem you can ignore the time).

For example, the file below shows several lines in the log file. Each line of the log file represents the netid of a user, the IP address of a specific web server, and the specific time the visit was recorded. Such a log file represents one day of network traffic at NYU.

**weblog.txt**

```
ab12:152.3.215.24-09/May/2024:10:30:45
cd34:68.71.216.171-09/May/2024:11:15:20
ab12:68.71.216.171-09/May/2024:12:00:05
ef56:199.239.136.200-09/May/2024:12:30:10
ef56:199.239.136.200-09/May/2024:13:45:55
ef56:98.142.99.33-09/May/2024:14:20:30
gh78:208.179.31.37-09/May/2024:15:00:12
ab12:152.3.215.24-09/May/2024:15:30:45
gh78:208.179.31.37-09/May/2024:16:10:20
cd34:152.3.215.24-09/May/2024:17:00:55
gh78:68.71.216.171-09/May/2024:18:20:30
ab12:152.3.215.24-09/May/2024:19:40:15
gh78:152.3.215.24-09/May/2024:20:15:00
```

The data above shows that the person with netid *ab12* visited website with IP address 152.3.215.24 three times, and website with IP address 68.71.216.171 one time. Website 68.71.216.171 was visited by the three users with netids *cd34*, *ab12*, and *gh78*.

You will write six (6) functions in this problem. Data samples and expected outputs are provided.

You should follow the following guidelines:

1. Choose variable names that describe **in plain English** what the variables represent.
2. You can answer a question by calling any of the **functions defined in previous questions**.
3. Add **as many comments** to make your code easy to understand for the grader.
4. When writing programs with *if - elif - else / while / for / def* statements, make very clear your **line indentations** (use a big enough indentation width). If indentation is not perfectly clear, the grader will consider "no indent".
5. The use of concepts that were not covered in class is **prohibited**.

1) Write a Python function named *processData* that has one string parameter that represents a datafile in a format similar to *weblog.txt*, and returns a list of three (3) items of information as follows:

   - a string that represents the netid of a user,
   - a string that represents the IP address visited by the user, and
   - a string representing the timestamp of the visit.

   We give two examples of calls to this function.

| | |
|---|---|
| **Call** | `processData("ab12:152.3.215.24-09/May/2024:10:30:45")` |
| **Returned value** | `['ab12', '152.3.215.24', '09/May/2024:10:30:45']` |
| **Call** | `processData("cd34:68.71.216.171-09/May/2024:11:15:20")` |
| **Returned value** | `['cd34', '68.71.216.171', '09/May/2024:11:15:20']` |

```
def processData(line):
```

2) Write a Python function named *fileToList* that has one string parameter that represents a datafile in a format similar to *weblog.txt* and returns a list of lists where each sublist has three (3) pieces of information representing one line from the datafile:

- a string that represents the netid of a user,
- a string that represents the IP address visited by the user, and
- a string representing the timestamp of the visit.

| | |
|---|---|
| **Call** | `fileToList("weblog.txt")` |
| **Returned value** | `[['ab12', '152.3.215.24', '09/May/2024:10:30:45'],`<br>` ['cd34', '68.71.216.171', '09/May/2024:11:15:20'],`<br>` ['ab12', '68.71.216.171', '09/May/2024:12:00:05'],`<br>` ['ef56', '199.239.136.200', '09/May/2024:12:30:10'],`<br>` ['ef56', '199.239.136.200', '09/May/2024:13:45:55'],`<br>` ['ef56', '98.142.99.33', '09/May/2024:14:20:30'],`<br>` ['gh78', '208.179.31.37', '09/May/2024:15:00:12'],`<br>` ['ab12', '152.3.215.24', '09/May/2024:15:30:45'],`<br>` ['gh78', '208.179.31.37', '09/May/2024:16:10:20'],`<br>` ['cd34', '152.3.215.24', '09/May/2024:17:00:55'],`<br>` ['gh78', '68.71.216.171', '09/May/2024:18:20:30'],`<br>` ['ab12', '152.3.215.24', '09/May/2024:19:40:15'],`<br>` ['gh78', '152.3.215.24', '09/May/2024:20:15:00']]` |

```
def fileToList(logfile):
```

3) Write a Python function named *userData* which has one string parameter that represents a datafile in a format similar to *weblog.txt* and that returns a dictionary in which:

- The keys are each unique netid stored in the logfile.
- The value associated to each key is a set of unique IP addresses visited by the person with that netid.

For example, for the key *ab123* the associated value is the set:

```
set(['68.71.216.171', '152.3.215.24'])
```

since although *ab123* visited a website four times, three of the visits were to the same website. For the data file shown above, the returned dictionary is printed below:

```
{'ab12': {'68.71.216.171', '152.3.215.24'},
 'cd34': {'68.71.216.171', '152.3.215.24'},
 'ef56': {'98.142.99.33', '199.239.136.200'},
 'gh78': {’68.71.216.171', '208.179.31.37', '152.3.215.24'}}
```

```
def userData(logfile):
```

4) Write a Python function named *ipVisited* has one string parameter that represents a datafile in a format similar to *weblog.txt* and that returns a dictionary in which:
   - The keys are each unique IP address stored in the file.
   - The value associated to each key is the total number of visits to the website with that IP address (including repeated visits to same site by same user).

   For the data file shown above, the returned dictionary is printed below:

```
{'152.3.215.24': 5,
 '68.71.216.171': 3,
 '199.239.136.200': 2,
 '208.179.31.37': 2,
 '98.142.99.33': 1}
```

   The dictionary should be sorted such that the first pair corresponds to the IP address with the highest number of visits and the last pair to the IP address with the fewest visits. Breaking ties doesn't matter.

   The IP address 152.3.215.24 is associated with value 5 since it was visited 3 times by user *ab12* and one time by users *cd34* and *gh78*, respectively.

```
def ipVisited(logfile):
```

5) Write a Python function named *getUniqueVisits* which has one string parameter that represents a datafile in a format similar to *weblog.txt* and that returns a dictionary in which:

- The keys are IP addresses of each unique website in the logfile.
- The value associated with the key is the list of unique netids of the users that visited the website with that IP address (unique means that each value in the list appears once).

For the logfile shown above the dictionary returned is shown below:

```
{'152.3.215.24': ['cd34', 'ab12', 'gh78'],
 '68.71.216.171': ['cd34', 'ab12', 'gh78'],
 '199.239.136.200': ['ef56'],
 '98.142.99.33': ['ef56'],
 '208.179.31.37': ['gh78']}
```

The order of the IP addresses in each list doesn't matter. Note that netid *ab12* appears once in the list associated with key *152.3.215.24* although that user visited the website three times as recorded in the log file.

```
def getUniqueVisits(logfile):
```

6) Write a Python function named *mostActive* has one string parameter that represents a datafile in a format similar to *weblog.txt* and that returns the netid of the user who visited the most different websites as recorded in the logfile shown at the beginning of this problem. Breaking ties doesn't matter: The function can return the netid of any of the most active users.

For that file, *mostActive* should return netid *gh78*, as that user visits 3 unique websites, which is more than any other user.

```
def mostActive(logfile):
```

*Extra page you can use as scratch paper*