CSE 160 Spring 2023 - Midterm Exam

Instructions:
- You have 40 minutes to complete this exam.
- The exam is closed book, including no calculators, computers, phones, watches or other electronics.
- You are allowed a single sheet of notes for yourself.
- We also provide a syntax reference sheet.
- Turn in all sheets of this exam, together and in the same order when you are finished.
- When time has been called, you must put down your pencil and stop writing.
  - Points will be deducted if you are still writing after time has been called.
- You may only use parts and features of Python that have been covered in class.
- All questions assume Python version 3.7, as we have been using all quarter.
- You may ask questions by raising your hand, and a TA will come over to you.

Good luck!

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td></td>
</tr>
<tr>
<td>Question 2</td>
<td></td>
</tr>
<tr>
<td>Question 3</td>
<td></td>
</tr>
<tr>
<td>Question 4</td>
<td></td>
</tr>
<tr>
<td>Question 5</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
</tr>
</tbody>
</table>
1. [6 pts] For each of the below expressions, write what the expression evaluates to and the type of that value. You should assume that variables have been declared and assigned as follows:  

```plaintext
a = 11  b = "hello"  c = False  d = 10.5  e = 2  f = { "a": 1, 1: "a", "b": b }
```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Evaluation</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 / 2</td>
<td>5.5</td>
<td>int</td>
</tr>
<tr>
<td>2 + 11</td>
<td>13</td>
<td>int</td>
</tr>
<tr>
<td>11 % 2 == 0 or not False</td>
<td>True</td>
<td>bool</td>
</tr>
<tr>
<td>not(len('hello') &lt; 5 and 10.5 &gt; 2)</td>
<td>True</td>
<td>bool</td>
</tr>
<tr>
<td>'hello'[1] + str(d * e)</td>
<td>'h10ello'</td>
<td>str</td>
</tr>
<tr>
<td>a = 10</td>
<td>10</td>
<td>int</td>
</tr>
<tr>
<td>b = 'hello'</td>
<td>'hello'</td>
<td>str</td>
</tr>
<tr>
<td>{&quot;a&quot;: 1, 1: &quot;a&quot;, &quot;b&quot;: b}['b'][2] + f[1]</td>
<td>21.5</td>
<td>int</td>
</tr>
</tbody>
</table>
2. Write a function `tricky_function(arr)` that modifies a given list in the following way: at index i, if the value at index i is divisible by 3, then the function should multiply the value at index i by its index; otherwise leave the value as it is.

Examples:

`tricky_function ([0, 1, 2, 3, 4, 5, 6, 7, 8]) = [0, 1, 2, 9, 4, 5, 36, 7, 8]`

`tricky_function ([2, 1, 21, 7, 4, 5, 4, 7, 18]) = [2, 1, 42, 21, 4, 5, 4, 7, 144]`

`tricky_function ([2, 3, 2]) = [2, 3, 2]`

`tricky_function ([1, 1, 1, 1]) = [1, 1, 1, 1]`
3. Write a function `join` that, given two lists of characters `c1` and `c2`, returns the number of pairs in `c1` and `c2` whose values are equal.

Examples:

\[
\begin{align*}
\text{join } ([a,b,c,e], [a,a,e,b,c,c,d]) &= 6 \\
&= \{[[a,a], [a,a], [b,b], [c,c], [c,c], [e,e]] \}
\end{align*}
\]

\[
\begin{align*}
\text{join } ([a,a], [a,a]) &= 4 \\
&= \{[[a,a], [a,a], [a,a], [a,a]] \}
\end{align*}
\]

\[
\begin{align*}
\text{join } ([a,b,c], [a,b,c]) &= 3 \\
&= \{[[a,a], [b,b], [c,c]] \}
\end{align*}
\]

\[
\begin{align*}
\text{join } ([x,y,z], [a,a,e,b,c,c,d]) &= 0 \\
&= \{
\}
\end{align*}
\]
4. After running this snippet of code, what will this function print?

```python
def foo(n1, n2, n3, b, s, d):
    if n1 % 2 == 0 or not b:
        print("A")
    elif str(n2 * n3) == 21.0:
        print("B")
    else:
        print("C")
    if d["b"][2] + d[1] == LA:
        print("D")
    if d["a"] == n1 - n2:
        print("E")
    else:
        print("F")

a = 11;
b = "hello";
c = False;
d = 10.5;
e = 2;
f = { "a": 1, 1: "a", "b": b }

foo(a, d, e, c, b, f)
foo(17, 21, 1, true, "hallo", {"a": 1, 1: "a", "b": "hello"})
foo(0, 3, 7.0, false, "hallo",{"a": -3, 1: "a", "b": "hallo"})
```
5. Write a function `max_consecutive_sum` that, given a list of integers, returns the absolute value of the largest sum between two consecutive numbers in the list. Return -1 if the length of the list is less than or equal to 1.

Examples:
```python
def max_consecutive_sum(nums):
    if len(nums) < 2:
        return -1
    max_sum = 0
    current_sum = 0
    for num in nums:
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)
    return max_sum
```

max_consecutive_sum([36, 45, 7]) = 81
max_consecutive_sum([3]) = -1
max_consecutive_sum([2, 3, 2]) = 5
max_consecutive_sum([1, 1, 1, 1]) = 2