4. Objects and Functions

Consider a Person class with the attributes

- name: a string representing the name of this person
- friends: a (possibly empty) list of Person objects representing this person's friends

(a) [10 points] Implement the following function according to the specifications. Your implementation must make effective use of range() in a for-loop.

**Hint:** Recall the Python keyword `in`, which returns `True` if a value is in a sequence, and `False` otherwise. For example, `2 in [2, 3, 4]` evaluates to `True`, but `5 in [2, 3, 4]` evaluates to `False`.

```python
def common(f1, f2):
    """Returns: a string list containing the names of the people that are in both Person list f1 and Person list f2.

    Example: Let p1, p2, ..., p6 be Person objects. If f1 is the list [p2, p3, p5] and f2 is the list [p3, p4, p6, p5], then common(f1, f2) returns a list containing the names of p3 and p5 (not p3 and p5 themselves).

    Precondition: f1 and f2 are each a nonempty list of Person objects.
    """

    namesList = []
    for k in range(len(f1)):
        p = f1[k]
        if p in f2:
            namesList.append(p.name)

    return namesList
```
(b) [5 points] Implement function `mutual_friends` according to the specifications below. Your implementation must use function `common` from part (a) in a meaningful way. Assume `common` has been correctly implemented. Pay attention to the specifications of both `mutual_friends` and `common`.

```python
def mutual_friends(p1, p2):
    """Returns: a string list containing the names of the mutual friends of Persons p1 and p2. If p1 and p2 have no mutual friends, return an empty list.
    """

    precondition: p1 and p2 are each a Person object.
    ""

    if p1.friends == [] or p2.friends == []:
        return []
    return common(p1.friends, p2.friends)
```
(c) [9 points] Implement the following function according to the specifications below. Your implementation must use a “for-each” loop meaningfully, i.e., you cannot use range() in your loop.

```python
def nickname_friends(p):
    """Returns: the number of names modified. This function modifies Person p's friends list such that the names longer than 5 characters will will be truncated to the first 5 characters and a "u" is appended. Names 5 characters in length or shorter remain unchanged.

    Example: If p has 3 friends named "Jonathan", "Benji", and "Tristan", then their names will become "Jonatu", "Benji" (unchanged), and "Tristu", respectively, and the function returns 2.

    Precondition: p is a Person object with a nonempty friends list.
    """
    changes = 0
    for friend in p.friends:
        if len(friend.name) > 5:
            changes += 1
            friend.name = friend.name[:5] + 'u'
    return changes
```

```python
changes = 0
for friend in p.friends:
    frame = friend.name
    if len(frame) > 5
        changes += 1
        frame = frame[:5] + 'u'
return changes
```