CS 111 Midterm 1

Junhui Cho

TOTAL POINTS

38 / 40

QUESTION 1

1 Indexing and slicing 7 / 7

✓ **- 0 pts** Correct

QUESTION 2

2 List comprehensions 6 / 8

✓ **- 2 pts** two or more issues in answer to part d

QUESTION 3

3 Tracing a recursive function 3 / 3

✓ **- 0 pts** Correct

QUESTION 4

4 Writing a function 1 5 / 5

✓ **- 0 pts** Correct

QUESTION 5

5 Tracing function calls 4 / 4

✓ **- 0 pts** Correct

QUESTION 6

6 Writing a function 2 5 / 5

✓ **- 0 pts** Correct

QUESTION 7

7 Writing a function 3 5 / 5

✓ **- 0 pts** Correct

QUESTION 8

8 Conditional execution 3 / 3

✓ **- 0 pts** Correct

~~U0464~~ U09465844

CS 111, Fall 2018 name: Junhui Cho email: jhoo @bu.edu

Midterm 1 lecture: ___ A1 (MWF 10:10) ___B1 (MWF 12:20) ___C1 (TuTh)

1. (7 points) Given the following assignment statements:

```
s = 'programmer'
values = [0, [2, 4], 6, 8, 10]
```

0 1 2 3 4 5 6 7 8 9

what is the value of each of the following expressions?

a. `s[4]` 'r'

b. `values[2]` 6

c. `values[-1:]` [10]

d. `s[4:-1]` 'ramme'

e. `values[::-2]`

[10, 6, 0]

2. (8 points) Evaluate each of the following:

0 1 2 3

a. `[3 + y for y in range(4)]`

[3, 4, 5, 6]

b. `[x[0] for x in ['bring', 'a', 'card'] if len(x) > 1]`

['b', 'c']

c. `[x for x in [1, 3, 5] if x > 2] + [2 for x in [1, 3, 5]]`

[3, 5, 2, 2, 2]

d. `[c == 'o' for c in 'donor']`

True

3. (3 points) Consider the following recursive function:

```
def mystery(a, b):
    if a < b:
        return b
    else:
        return a + mystery(a - 1, b + 2)
```

a. Trace the execution of the function call `mystery(6, 2)`. You may use any reasonable approach, provided that you show *all* of the recursive calls. You are ***not*** required to show the frames.

mystery(6, 2)
return 6 + mystery(5, 4) ⇒ 17

mystery(5, 4)
return 5 + mystery(4, 6) ⇒ 11

mystery(4, 6)
return 6

b. What is the value returned by `mystery(6, 2)`?

17

4. (5 points) Write a recursive function named `remove_spaces(s)` that takes a string `s` and returns a string consisting of the characters of `s` but with all spaces removed. For example, `remove_spaces('I love you')` should return the string `'Iloveyou'`, and `remove_spaces('  oh  no')` should return the string `'ohno'`. **Important:** You *must* use recursion. You may *not* use a list comprehension, or any construct or function that we have not discussed in lecture.

```
def remove_spaces(s):
    if s == '':
        return ''

    else:
        rest_s = remove_spaces(s[1:])
        if s[0] == ' ':
            return rest_s

        else:
            return s[0] + rest_s
```

email
jhoo@bu.edu

5. (4 points) What is the output of the following program?

Put the output below:

```
def foo(x, y):
    print('foo', x, y)
    x = x + y
    y = bar(x + 1)
    print('foo', x, y)
    return y

def bar(x):
    x = x - 3
    return x

x = 5
y = 3
x = foo(x, y)
print(x, y)
foo(y, y)
print(x, y)
```

foo 5 3
foo 8 6
6 3
foo 3 3
foo 6 4
6 3

6. (5 points) Write a function `min_pos(values)` that takes a list of numbers called `values` and returns the ***smallest positive*** value from that list. Values less than or equal to 0 should ***not*** be returned. For example:

- `min_pos([-2, 6, 3, 8, 0])` should return 3
- `min_pos([-2, 6, -3, 8, 0])` should return 6

Your function should use either recursion *or* a list comprehension. You are also welcome to use one or more of the built-in functions that we have discussed in lecture. You may assume that the list contains at least one positive number.

```
def min_pos(values):
    pos_vals = [x for x in values if x > 0]
    return min(pos_vals)
```

7. (5 points) Write a recursive function named `num_occur(val, values)` that takes as inputs a single value `val` and a list `values` containing 0 or more elements and that returns the number of times that `val` occurs in `values`. For example:

- `num_occur(7, [7, 7, 7, 5, 3, 5])` should return 3
- `num_occur(5, [7, 7, 7, 5, 3, 5])` should return 2
- `num_occur(3, [7, 7, 7, 5, 3, 5])` should return 1
- `num_occur(8, [7, 7, 7, 5, 3, 5])` should return 0

Important: You *must* use recursion. You may *not* use a list comprehension, or any construct or function that we have not discussed in lecture.

```
def num_occur(val, values):
    if len(values) == 0:
        return 0
    else:
        rest_values = num_occur(val, values[1:])
        if values[0] == val:
            return rest_values + 1
        else:
            return rest_values
```

8. What is the output of the following Python program? (3 points)

```
val = 7
if val > 10 or val < 15:
    print('fun')
    if val ** 2 > 25:
        print('fad')
if val == 7:
    print('fit')
elif val < 20:
    print('for')
else:
    if val % 2 == 1:
        print('fee')
```

Put the output below:

fun
fad
fit

email
jhoo@bu.edu

Blank page for extra work;
do not detach!

def min_pos(values)