Midterm 3 Review: CSE216 –Programming Abstractions

Section 1: Python

1. What is the output of following Python codes which uses Lambda expressions?

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

print(new_list)
```

```
[4, 6, 8, 12]
```

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

```
[2, 10, 8, 12, 16, 22, 6, 24]
```

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

expr1 = lambda data: float(sum(data)/len(data))

print(expr1(my_list))
```

```
6.25
```

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

expr2 = lambda data: ((data[len(data)-1] - data[0])/data[0] * 100)

print(expr2(my_list))
```

```
1100.0
```

Create a Python file recursion.py and write recursive Python functions along with test cases for the following tasks:

```
# 3.1 Computes the sum of all the even elements in the list u.
# sum_evens([1, 2, 3, 4] returns 6
# sum_evens([1, 2, 3, 4, 5, 6, 7, 8, 9, 10] returns 30
#
def sum_evens(u):
    return None  # Replace this with your implementation


# 3.2 Finds the even elements in the list u and returns their indices as a list.

# find_even_indices([1, 2, 3, 4] returns [1, 3]
# find_even_indices([1, 2, 3, 4, 5, 6, 7, 8, 9, 10] returns [1, 3, 5, 7, 9]
# Note how i parameter is used in this design.
#
def find_even_indices(u):
```

```
    return find_even_indices_aux(u, 0)

def find_even_indices_aux(u, i):
    return None  # Replace this with your implementation


# 3.3 Returns the zip of two strings s1 and s2 of the same length.
# zips('ftp', 'abc') returns 'fatbpc'
#
def zips(s1, s2):
    return None  # Replace this with your implementation

# 3.4 Finds and returns only the vowels in the string s in the same order they appear in
s.
# find_vowels('ftp') returns ''
# find_vowels('Apple') returns 'Ae'
# find_vowels('Banana Republic') returns 'aaaeui'
#
def find_vowels(s):
    return None  # Replace this with your implementation


# 3.5 Finds the vowels in a string s and returns their indices as a string.
# find_vowel_indices('Apple') returns '04'
# find_vowel_indices('Banana Republic') returns 13581013
# Note how i parameter is used in this design.
#
def find_vowel_indices(s):
    return find_vowel_indices_aux(s, 0)

def find_vowel_indices_aux(s, i):
    return None  # Replace this with your implementation
```

Section 2: Java

2. Consider the following Java source code to answer questions in this section. What will be the value of players array after each sorting operation?

```
import java.util.Arrays;
import java.util.Comparator;

public class LambdaSorting {
    public static void main(String[] args) {

        String[] players = {"Jung Bong", "DaeSung Ko", "Seunghwan Oh", "Heeseop Choi",
        "Tommy Phelps", "Chong Taehyon"};

        Arrays.sort(players, (String s1, String s2) -> (s1.compareTo(s2)));

          Arrays.asList(players).forEach((player) -> System.out.println(player));
```

```
Chong Taehyon
DaeSung Ko
Heeseop Choi
Jung Bong
Seunghwan Oh
```

```
Tommy Phelps

        Arrays.sort(players, (String s1, String s2) -> (s1.substring(s1.indexOf("
        ")).compareTo(s2.substring(s2.indexOf(" "))));

        Arrays.asList(players).forEach((player) -> System.out.println(player));

Jung Bong
Heeseop Choi
DaeSung Ko
Seunghwan Oh
Tommy Phelps
Chong Taehyon

        Arrays.sort(players, (String s1, String s2) -> (s1.length() - s2.length()));

        Arrays.asList(players).forEach((player) -> System.out.println(player));

Jung Bong
DaeSung Ko
Heeseop Choi
Seunghwan Oh
Tommy Phelps
Chong Taehyon

        Arrays.sort(players, (String s1, String s2) -> (s1.charAt(s1.length() - 1) -
        s2.charAt(s2.length() - 1)));

        Arrays.asList(players).forEach((player) -> System.out.println(player));

Jung Bong
Seunghwan Oh
Heeseop Choi
Chong Taehyon
DaeSung Ko
Tommy Phelps
    }

}
```

Section 3: SML

3. Given the following function definition:

```
fun double x = 2 * x;
fun triple x = 3.0 * x;
```

What is the result of `double(triple(3.0));`

a) `18.0`
b) `18`
c) `Error: operator and operand do not agree`
d) `double(9)`

4. Select the expression which will evaluate to [1,2,3,4]:

a) `1::[2,3]@[4]`
b) `[1]::[2,3]@[4]`

c) `explode "1234"`
d) `[1,2,3]::4`

5. Given the following recursive function definition select the value of f(2,5):

```
fun f(a, 0) = 1
    |   f(a, n) = a * f(a, n-1);
```

a) `0 : int`
b) `10 : int`
c) `32 : int`
d) `25 : int`

6. Select the recursive function which defines the sequence 5, 8, 11, 14, 17, 20 .. when executed on inputs 0,1,2,3,4,5…

a)
```
fun s(0) = 5
  |   s(n) = n + 3;
```

b)
```
fun s(0) = 5
  |   s(n) = 3*n + 5
```

c)
```
fun s(n) = 3 * n + 5;
```

d)
```
fun s(0) = 5
  |   s(n) = 3 + s(n-1)
```

7. Give the type of the following function:

```
fun g x a b = 2*a + 3*b;
```

a) `'a -> int -> int -> int`
b) `int -> int -> int`
c) `('a * int * int) -> int`
d) `int -> int -> int -> int`

8. Given the following tuple, what is the output of `#1(#2(tuple));`?

```
val tuple = (4,(5.0,6),"abcd",("e","f"));
```

a) `(5.0, 6)`
b) "a"
c) `5.0`
d) "abcd"

9. Select the recursive function which defines the sequence "and his dog", "1 man and his dog""2 man 1 man and his dog", "3 man 2 man 1 man and his dog",…

a)
```
fun medow(0) = "and his dog went to mow a medow."
  |   medow(n) = Int.toString n ^
  (if n>1 then " men " else " man ") ^ medow(n-1);
```

b)
```
fun medow(0) = "and his dog"
  |   medow(n) = n ^ " man " ^ medow(n-1);
```

c)
```
fun medow(0) = "and his dog"
  |   medow(n) = Int.toString n ^ " man " ^ medow(n-1);
```

```
d) fun medow(0) = " and his dog"
   |   medow(n) = makestring n ^ " man" ^ medow(n-1);
```

1. Given the following function definitions:

```
fun double x = 2 * x;
fun triple x = 3 * x;
```

What is the result of `double(triple(size "seven"));`

a) `42 : int`
b) `"fourty two"`
c) `30 : int`
d) `25 : int`

10. Define the following function:

```
count_1s [4,3,1,6,1,1,2,1] = 4
```

```
fun count_1s nil     = 0
|   count_1s (1::t) = 1 + count_1s t
|   count_1s (h::t) = count_1s t;
```

11. Given a function map as follows determine the output of expressions given below:

```
Fun map f nil = nil (* pre-defined anyhow *)
| map f (h::t) = (f h)::map f t;
```

a) `map(fn s => s^"io") ["pat", "stud", "rat"];`

`["patio","studio","ratio"]____`

b) `map(hd o explode)["final","omega","previous","persist"];`

`__[#"f",#"o",#"p",#"p"]_____________________________`

tasks:

4.1 Write a function reverse : 'a list -> 'a list to reverse a list.

```
- reverse ["a", "b", "c"];
val it = ["c","b","a"]
```

4.2 Write a function compress to remove consecutive duplicates from a list.

```
-compress ["a","a","a","b","c","c","a","a"];
val it = ["a","b","c","a"] : string list
```

4.3 Write a function cluster that uses accumulator to cluster consecutive duplicate into nested lists.

```
-cluster(["a","a","a","b","c","c","a","a"], []);
val it = [["b"],["a","a","a"],["c","c"],["a","a"]] : string list list
```

4.4 Define a function sumlists: int list * int list -> int list which takes in input two lists of integers and gives as result the list of the sums of the elements in corresponding position in the input lists. The shortest list has to be seen as extended with 0's. Examples:

```
sumlists([],[]) = []
sumlists([1,2],[3,4]) = [4,6]
sumlists([1],[3,4,2]) = [4,4,2]
sumlists([1,6],[3]) = [4,6]
```

4.5 Define a function flatten: 'a list list -> 'a list which takes in input a list of lists and gives back the list consisting of all the elements, in the same order in which they appear in the argument. Examples:

```
flatten [] = []
flatten [[]] = []
flatten [[1,2],[2,3,4],[5],[],[6,7]] = [1,2,2,3,4,5,6,7]
flatten [["a"],["b","a"]] = ["a","b","a"]
```

Python recursion answers:

```python
def sum_evens(u):
    if (u == []):
        return 0
    else:
        if (u[0] % 2 == 0):
            return u[0] + sum_evens(u[1:])
        else:
            return sum_evens(u[1:])

def find_even_indices_aux(u, i):
    if (u == [] or i >= len(u)):
        return []
    else:
        if (u[i] % 2 == 0):
            return [i] + find_even_indices_aux(u, i + 1)
        else:
            return find_even_indices_aux(u, i + 1)

def find_even_indices(u):
    return find_even_indices_aux(u, 0)

def zips(s1, s2):
    if(len(s1) > len(s2)):
        if(len(s2) > 1):
            return s1[0] + s2[0] + zips(s1[1:], s2[1:])
        else:
            return s1 + s2
    else:
        if (len(s1) > 1):
            return s1[0] + s2[0] + zips(s1[1:], s2[1:])
        else:
            return s1 + s2

def find_vowels(s):
    if (len(s) == 0):
        return ''
```

```
    else:
        if (s[0].lower() in ['a', 'e', 'i', 'o', 'u']):
            return s[0] + find_vowels(s[1:])
        else:
            return find_vowels(s[1:])

def find_vowel_indices_aux(s, i):
    if (len(s) == 0 or i >= len(s)):
        return ''
    else:
        if (s[i].lower() in ['a', 'e', 'i', 'o', 'u']):
            return str(i) + find_vowel_indices_aux(s, i + 1)
        else:
            return find_vowel_indices_aux(s, i + 1)

def find_vowel_indices(s):
    return find_vowel_indices_aux(s, 0)


if __name__ == "__main__":
    print("sum_evens([1, 2, 3, 4]) = {}".format(sum_evens([1, 2, 3, 4])))
    print("sum_evens([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) = {}".format(sum_evens([1, 2, 3,
4, 5, 6, 7, 8, 9, 10])))

    print("find_even_indices([1, 2, 3, 4]) = {}".format(find_even_indices([1, 2, 3,
4])))
    print("find_even_indices([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) =
{}".format(find_even_indices([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])))

    print("zips('ftp', 'abc') = {}".format(zips('ftp', 'abc')))

    print("find_vowels('ftp') = {}".format(find_vowels('ftp')))
    print("find_vowels('Apple') = {}".format(find_vowels('Apple')))
    print("find_vowels('Banana Republic') = {}".format(find_vowels('Banana Republic')))

    print("find_vowel_indices('Apple') = {}".format(find_vowel_indices('Apple')))
    print("find_vowel_indices('Banana Republic') = {}".format(find_vowel_indices('Banana
Republic')))
```

SML program answers:

```
fun reverse_aux (lst, r) = if lst = nil then r else reverse_aux(tl(lst), hd(lst)::r)

fun reverse (lst) = reverse_aux (lst, []);

fun compress_aux (lst, d) = if lst = nil then nil else if hd(lst) = d then compress_aux
(tl(lst), d)
                            else hd(lst)::compress_aux (tl(lst), hd(lst));

fun compress (lst) = hd(lst)::compress_aux (tl(lst), hd(lst));

fun cluster(lst, accum) = if lst = nil then accum
else if accum = nil then cluster(tl(lst), [hd(lst)]::[])
else if hd(lst) = hd(hd(accum)) then cluster (tl(lst), (hd(lst)::hd(accum))::tl(accum))
else cluster (tl(lst), [hd(lst)]::(hd(accum)::tl(accum)));
```

```
fun sumlists (lst1, lst2) = if lst1 = nil andalso lst2 = nil then nil else if lst1 = nil
then lst2 else if lst2 = nil then lst1 else hd(lst1)+hd(lst2)::sumlists(tl(lst1),
tl(lst2));


fun flatten(lst) = if lst = nil then nil else hd(lst) @ flatten(tl(lst));

reverse ["a", "b", "c"];
compress ["a","a","a","b","c","c","a","a"];
cluster(["a","a","a","b","c","c","a","a"], []);
sumlists([],[]);
sumlists([1,2],[3,4]);
sumlists([1],[3,4,2]);
sumlists([1,6],[3]);
flatten [];
flatten [[]];
flatten [[1,2],[2,3,4],[5],[],[6,7]];
flatten [["a"],["b","a"]];
```