Q#	1	2	3	4	5	6	7	8	**Total**
Marks	15	8	5	6	6	6	5	7	**58**

Instructions (Read carefully before the exam begins):

1. Before you begin, make certain that you have one Exam Booklet with 15 pages, and one double-sided Reference Sheet.
2. All coding is to be completed using **Python 3**.
3. See the Reference Sheet for information about useful Python language features. You may use language features not listed on the Reference Sheet, as long as they were discussed in modules 01 through 05 and were allowed on assignments 01 through 05. In particular, **loops are not allowed**.
4. Include only design recipe elements that are explicitly requested. In particular, most questions require you only write the function definition (body).
5. Otherwise follow the CS 116 style guidelines for your solutions.
6. Unless indicated otherwise, you may use a helper function or `lambda` anywhere you feel it is needed. Helper functions should be defined separately from required functions (either before or after is acceptable). Unless otherwise stated, helper functions also do not need design recipe elements other than the body.
7. If you require more space to answer a question, there is a blank page at the end you may use instead, but you must clearly indicate in the provided answer space that you have done so.
8. Read each question carefully, as many demand a specific approach or specify specific restrictions.
9. **Do not detach any pages and do not write on the QR codes**

Q1. **[15 marks]** Each box presents a Python code fragment and a question. In the box on the right, answer the question, or write `ERROR` if the code results in an error. Each code snippet is independent, and variables and functions are not shared between them.

Read the question carefully: Most ask for the value of a variable, but some ask for what is printed. Double-check your answer, as many questions are based on common sources of bugs!

When the value of a variable is asked for, make sure the type is clear: If it's a String, put it in quotes. If it's a Float, make sure it has a decimal part even if it's .0, and if it's an Int, make sure it does not end in .0. If it's a List, put it in square brackets.

When the printed output is asked for, write exactly what is printed. Do not add quotes unless quotes are actually printed.

Code	Answer

(a) [1 mark] What is the value of `x` after running this code?
 	
`x = 1`
`z = 2`
`y = z`
`z = 3`
`x = y`
	x = 2

(b) [1 mark] What is the value of `n` after running this code?
 	
`n = 0`
`if 2*n >= n:`
`  n = 1`
`elif 2*n < n:`
`  n = 2`
`else:`
`  n = 3`
	n = 1

(c) [1 mark] What is the value of `c` after running this code?
 	
`s = "good day"`
`t = s[2:6]`
`u = t.split()`
`c = "".join(u)`
	c = "odd"

(d) [1 mark] What is the value of `z` after running this code?
 	
`a = True or False`
`b = True and False`
`z = 0`
`if a or b:`
`  z = 1`
`if b == (not a):`
`  z = 2`
	z = 2

(e) [1 mark] What is the value of `b` after running this code?
 	
`x = 5 % 3`
`y = 5 // 3`
`b = x + y`
	b = 3

(f) [1 mark] What is the value of `x` after running this code?
 	
`def f(t):`
`  x = t + 1`
 	
`x = f(5)`
	x = None

(g) [1 mark] What is the value of `y` after running this code?
 	
`def g(L):`
`  L[0] = 0`
`  L[-1] = L[0]`
`  return sum(x)`
 	
`x = [3,4,5]`
`y = g(x)`
	y = 4

(h) [1 mark] What is printed after running this code?
 	
`def f(n):`
`  if n == 1:`
`  return 0`
`  else:`
`  print(n)`
`  return f(n-1)`
 	
`f(3)`
	3
	2

(i) [1 mark] What is the value of `P` at the end of this code?
 	
`L = [0,2,4]`
`M = [1,3,5]`
`P = list(map(lambda x: x % 2,L + M))`
	P = `[0,0,0,1,1,1]`

(j) [1 mark] What is the value of `s` after running this code?
 	
`s = "tape"`
`t = 2*s`
`if "pet" in t:`
`  s = "jump"`
`if "pet" in s:`
`  s = "cry"`
`else:`
`  s = "sleep"`
	s = `"sleep"`

(k) [1 mark] What is the value of `t` after running the code below?
 	
`t = "banana"`
`t.replace("a","o")`
`t = t[1] + t[-1]`
	t = `"aa"`

(l) [1 mark]
What is returned after running `f(3)`?
 	
`def g(a,n):`
`  if n == 0:`
`  return 1`
`  return a + g(a,n-1)`
 	
`def f(k):`
`  return g(k,k)`
	`10`

(m) [1 mark] What does this code print?
 	
`def talk(L):`
`  if L != []:`
`  print(L.pop())`
`  talk(L)`
`  return 0`
 	
`talk(["w","e", "p"])`
	`p`
	`e`
	`w`

(n) [1 mark] What does this code print?
 	
`def f(a,b,c):`
`  s = "{1} + {2} = {0}".format(b,a,c)`
`  return s`
 	
`eq = f(1,2,3)`
`print(eq)`
	`1 + 3 = 2`

(o) [1 mark]
What is the value of `x` after running this code?
 	
`x = 0`
`if print("hello") == "hello":`
`  x = 5`
`elif "hello" == "hello"[:]:`
`  x = 10`
`else:`
`  x = 20`
	`x = 10`

Q2. [8 marks] Read the following carefully:

- All parts of this question must be solved using **abstract list functions.**
- You must **not** use recursion.
- You must **not** define any additional helper functions.
- You may use `lambda`, as well as list and string functions.

[2] (a) The function `clean_up` consumes a list of strings `L` and returns a list similar to `L` (same order of strings) where all strings containing the character `'!'` are removed. Examples:

```
clean_up([]) => []
clean_up(['Boo!']) => [ ]
clean_up(['Oh', 'No!oo!', 'My', 'Dog!!']) => ['Oh', 'My']
```

Complete the function body for `clean_up`.

```
def clean_up(L):

    return list(filter(lambda s: '!' not in s,L))
```

[2] (b) The function `add_index` consumes a list of integers. For each element, it adds the index of the element to the element itself and returns the new list. Examples:

```
add_index([]) => []
add_index([5]) => [5]
add_index([4, 0, -8]) => [4, 1, -6]
```

Complete the function body for `add_index`.

```
def add_index(L):

    return list(map(lambda i: L[i] + i, range(len(L))))
```

[2] (c) The function `remove_duplicates` consumes a list of strings, L, and returns a list similar to L (same order of strings) without duplicated strings. You must NOT mutate the original list. Examples:

```
remove_duplicates([]) => []
remove_duplicates(['a']) => ['a']
remove_duplicates(['a','bb','c','b','a','a']) => ['bb', 'c', 'b']
```

Complete the function body for `remove_duplicates`.

```
def remove_duplicates(L):
    return list(filter(lambda c: L.count(c) == 1,L))
```

[2] (d) The function `grow` consumes a positive natural number `n` and prints an increasing collection of the character `'x'` starting from 1 up to size `n` Examples:

```
grow(1) => None and prints
x
grow(5) => None and prints
x
xx
xxx
xxxx
xxxxx
```

Complete the function body for `grow`.

```
def grow(n):
    list(map(lambda i : print((i+1)*'x'), range(n)))
```

Q3. [5 marks] The function `add_nums` consumes a positive natural number `n` and prompts the user `n` times to `Enter a number:`. After the user enters a number `n` times it returns the total of all numbers.

For example `add_nums(3)` would generate the following interaction (assuming the user enters `2, 51` and `1`):

```
Enter a number: 2
Enter a number: 51
Enter a number: 1
```

and it would return 54.

[3] (a) Complete the body of `add_nums`. You may assume the user will only enter valid numeric input.

```
def add_nums(n):


    if n == 0:
        return 0
    else:
        v = input("Enter a number: ")
        return int(v) + add_nums(n-1)
```

[2] (b) Write a test for `add_nums` using all appropriate `check` functions. Choose any `n >= 2`

```
check.set_input("1","2","3")
check.expect("test",add_nums(3),6)
```

Q4. **[6 marks]** The function `process_mixed_list` consumes a list of strings, `lst`, and returns a list of length 3, where the first element in the returned list contains all the strings in `lst` (in the same order as in `lst`) that contain only digits, the second element in the returned list contains all strings in `lst` (in the same order as in `lst`) that contain only alphabetical characters, and the third element in the returned list contains the rest of the strings (in the same order as in `lst`). For example

```
process_mixed_list([]) => ([[],[],[]])
process_mixed_list(["wow", "CS116", "2018", "","0"])
    => [['2018', '0'], ['wow'], ['CS116', '']]
```

[1] (a) Write the contract for `process_mixed_list`

```
(listof Str) -> (list (listof Str) (listof Str) (listof Str))
```

[5] (b) Complete the body of `process_mixed_list` using **accumulative recursion**

```
def process_mixed_list(lst):
    return keep_acc(lst,[],[],[])

def keep_acc(lst,res1, res2, res3):
    if lst==[]:
        return [res1,res2,res3]
    elif lst[0].isdigit():
        return keep_acc(lst[1:],res1+[lst[0]], res2, res3)
    elif lst[0].isalpha():
        return keep_acc(lst[1:],res1, res2+[lst[0]], res3)
    else:
        return keep_acc(lst[1:],res1, res2, res3+[lst[0]])
```

Q5. [6 marks] Consider the two functions below:

```
def do_something(x, y):
    if y >= len(x):
        return None
    elif x[y] not in x[:y]+x[y+1:]:
        return x.pop(y)
    return do_something(x, y+1)


def do_something_else(x):
    v = do_something(x,0)
    x.insert(len(x),v)
```

[1] (a) What is returned when `do_something([1,5,1,3,3],0)` is called?

`5`

[1] (b) What is returned when `do_something([2,5,1,3,3],2)` is called?

`1`

[1] (c) Give an example of an `x` with `len(x) == 3` such that `do_something(x,0)` returns `None`

Any list (or string) with the same elements (e.g. `[2,2,2]` or `'aaa'`)

[2] (d) In your words what does the function `do_something(L,0)` do for a non-empty list `L`?

Returns the first unique item in the list `L`. The list `L` is mutated to no longer contain that item.

[1] (e) In your words what does the function `do_something_else` do?

Mutates `x` such that the first unique item is moved to the end. If there are no unique items then places `None` at the end.

Q6. [6 marks] Write the function `replace_unique` that takes a list,`L`, and replaces all unique values in L with `"*"`. For example:

if `L = [1,2,3,2,3,3,0,5,4,4]` then `replace_unique(L) => None` and `L` is mutated to become `['*', 2, 3, 2, 3, 3, '*', '*', 4, 4]`

if `L = []`, `replace_unique(L) => None` and `L` is not mutated.

[4] (a) Complete the body of `replace_unique`

```
def replace_unique(L):
    replace_by_index(L,0)


def replace_by_index(L, ind):
    if ind < len(L):
        if L[ind] not in L[:ind]+L[ind+1:]:
            L[ind]="*"
        replace_by_index(L,ind+1)


*** Alternative solution using do_something from Q5 ****

def replace_unique(L):
    replace_from(L,0)

def replace_from(lst,pos):
    if pos < len(lst):
        toRemove = do_something(lst.copy(),pos)
        if toRemove != None:
            index = lst.index(toRemove)
            lst[index] = '*'
            replace_from(lst,index+1)
```

[2] (b) Use the appropriate check functions to write a test for `replace_unique` when the input is `L = [1,2,2,3]`.

```
L = [1,2,2,3]
check.expect("test",replace_unique(L),None)
check.expect("test(L)",L,['*',2,2,'*'])
```

Q7. [7 marks]

A palindrome is a sequence of non-whitespace characters that read the same backwards as forwards (e.g. Dad, Mom, eye, a, bb) in a case-insensitive fashion.

[5] (a) On the next page complete the missing code block in the function `get_palindrome`. which consumes a string `s`. It identifies the palindrome **words** (i.e. any sequence of characters that does not contain whitespace) in the string `s` and prints them on the same line.

Each palindrome word is printed in uppercase (e.g. ANA) and separated from other palindromes by a single space.

Note: You are NOT allowed to create any global variables. You should use the given `is_palindrome` function and **NOT** create new helper functions.

```
def is_palindrome(word):
    '''
    returns True if word is a palindrome,
    and False otherwsie.

    is_palindrome: Str -> Bool

    Examples:
      is_palindrome("Ana") => True
      is_palindrome("b") => True
      is_palindrome("Victoria") => False
    '''
    return word.lower() == word[::-1].lower()


def get_palindrome(s):
    '''
    prints the palindrome words found in a string s on the same line.

    Effects: prints to the screen

    get_palindrome: Str -> None

    Examples:
      get_palindrome("") => None
      get_palindrome("aA ") => None and prints AA
      get_palindrome("B ") => None and prints B
      get_palindrome("   Ana saw Bob draw a tiger ") => None
       and prints
      ANA BOB A
       on the same line
    '''
```

(the body of the code is started below. Complete the rest of the function.)

```
    s = s.strip()
    if s == "":
        return None



    elif s.find(" ") == -1: # only one word exists

        if is_palindrome(s):
            print(s.upper())
        return None
    else:  # more than one word exist
        stopIndex = s.find(" ")
        if is_palindrome(s[:stopIndex]):
            print(s[:stopIndex].upper(), end=" ")
        return get_palindrome(s[stopIndex:])
### ALTERNATE SOLUTION #####
    words = s.split()
    pals = list(filter(is_palindrome,words))
    upper = list(map(lambda s: s.upper(),pals))
    print(" ".join(upper))
```

[2] (b) Write a test case for the function `get_palindrome` using the string
" `Ana saw Bob draw a tiger` " and all appropriate `check` functions.

```
s = "   Ana saw Bob draw a tiger "
check.set_print_exact('ANA BOB A')
check.expect("test get_palindrome", get_palindrome(s), None)
```

Q8. **[5 marks]** Write a function `replace_digit` that consumes a positive integer parameter `num`, a single digit integer (between 0-9 inclusive) parameter `old_digit`, and a single digit integer (between 0-9 inclusive) parameter `new_digit`. The function returns a new integer similar to `num` after replacing all the digits from `num` that equal `old_digit` with `new_digit`.

Note: **You cannot use strings or lists for this question**

For example

```
replace_digit(467272,7,5) => 465252
replace_digit(10571,1,0) => 570
replace_digit(444444,4,0) => 0
```

```
def replace_digit(num,old_digit,new_digit):


if num <= 9 and num != old_digit:
    return num

if num <= 9:
    return new_digit

if (num % 10) == old_digit:

    return replace_digit(num//10, old_digit, new_digit) * 10 + new_digit

return replace_digit(num//10, old_digit, new_digit) * 10 + num%10

### (Alternative solution) ###
def replace_digit(num,old_digit,new_digit):

    return build(num,old_digit,new_digit,1)

def build(num, old, new, val):
    if num // 10 == 0:
        if num == old:
            num = new
        return val*num
    else:
        digit = num % 10
        if digit == old:
            digit = new
        return val*digit + build(num // 10,old,new,val*10)
```

This page intentionally left blank. You may continue solutions to any question here, if additional space is needed. However, be sure to clearly indicate which question is answered here, and indicate in the original space for the question that it is answered or continued here.