CS303E Week 9 Worksheet: Lists

Name: ______________________________ EID: ______________

Read the questions carefully, and answer each question in the space provided. Use scratch paper to do your work and then copy your answers neatly and legibly onto the test paper. Only answers recorded on the test paper will be graded.

1. (11 points: 1 point each) The following are true/false questions. **Write either T or F in the boxes at the bottom of page 1.** If there's any counterexample, it's false.

   (a) Comparing two lists of different lengths will result in an error. Not true! Similar to how we can compare strings of different lengths.

   (b) It is possible to directly convert a list to a string in Python.
   Yes! We simply do str(listName). List has an __str__ method defined in its class.

   (c) The `append()` method adds an element to the beginning of a list in Python. Nope! Adds to the end

   (d) The expression `[x for x in range(10) if not x % 2]` creates a list containing only positive even integers. False! This list will contain [0, 2, 4, 6, 8], and 0 is not positive.

   (e) Lists cannot be negatively indexed.

   (f) The `remove()` method deletes an element at a specified index from a list.
   Close! remove() takes a single value as input, and removes the first match of that value in the list.

   (g) It is possible to convert items of other types directly to a list in Python.
   Yes! We can do something like list("goose") and get back ["g", "o", "o", "s", "e"].

   (h) Lists can only contain items of the same data type.

   (i) The `+=` operator in Python can be used to concatenate two lists.

   (j) Lists have a fixed maximum size, and once this size is reached, no more elements can be added to the list. Lists have no maximum size in Python.

   (k) The `pop()` method in Python is used to remove and return the last element of a list. True, but you can actually specify an index, so pop() could potentially return any element of the list. I should clarify this question. My bad =)

a	b	c	d	e	f	g	h	i	j	k
F	T	F	F	F	F	T	F	T	F	T

Page total: ________/11

Questions 2-8 are multiple choice. Each counts 2 points. **Write the letter of the BEST answer in the box on the next page. Please write your answer in UPPERCASE. Each problem has a single answer.**

2. What is the primary difference between the list methods `.append()` and `.extend()`?

2A) Not exactly true! .extend() is used to combine two lists. But the lists we combine don't need to have multiple elements. EX: lst1.extend([1])

   A. `.append()` is used to add a single element to the end of the list, while `.extend()` is used to add multiple elements to the end of the list.
   B. `.extend()` is used to add a single element to the end of the list, while `.append()` is used to add multiple elements to the end of the list. Wrong on both counts!
   C. `.append()` is used for merging two lists, while `.extend()` is used to add a single element to a list. Opposite is true
   D. `.extend()` is used for merging two lists, while `.append()` is used to add a single element to a list. Yes!
   E. Both methods are identical; they can be used interchangeably to achieve the same result. Nooooooooo!

3. Which of the following list comprehensions is not valid in Python?

   A. `[x for x in range(0)]` [] — range(0) is empty, so our resulting list will be empty.
   B. `[x**2 for x in range(5) if x % 2 == 0]` [0, 4, 16]
   C. `[len(item) for item in ["my", "anaconda", "don't"]]` [2, 8, 5]
   D. `[x if x > 0 else -1 for x in range(5)]` [-1, 1, 2, 3, 4]
   E. `[x for x in "mercury is in retrograde =("]` [“m”, “e”, “r”, “c”, “u”, “r”, “y”, … etc]
   F. All of the above are valid list comprehensions. Yes!

4. Assuming `list1` is a list, which of the following declarations will not create a new list object with the same items as `list1`?

   A. `list2 = list1[0:len(list1)]` Slicing returns a new list.
   B. `list3 = list1` This creates a variable called list3, but it will refer to the same list as list1.
   C. `list4 = [goodies for goodies in list1]` List comprehension creates a new list.
   D. `list5 = list(list1)` The list constructor creates a new list.
   E. All of the above create lists distinct from `list1` in memory.

   list3 will be the same object as list1.

5. Assuming `myLst = ["Basil", "Kookie", "KitKat", "Grey"]`, what does the expression `myLst[:-2]` evaluate to?

   A. `["Basil, "Kookie"]`
   B. `["Basil, "Kookie", "KitKat"]`
   C. `["Kookie, "KitKat", "Grey"]`
   D. Error

```
 -4        -3         -2        -1
["Basil", "Kookie", "KitKat", "Grey"]
```

myLst[:-2] asks us to get all items in myLst, up to (but not including) the item at index -2. So the answer will be [“Basil”, “Kookie”]

6. In Python, when comparing two lists using the greater-than (>) or less-than (<) operators, how is the comparison evaluated?

   A. The comparison checks if the lengths of the lists are greater than or less than each other. I would hope not! Then [100] would be less than [1, 1].

   B. The comparison checks if the elements of the lists, when converted to strings, are greater than or less than each other in lexicographical order.

   C. The comparison checks if the elements at corresponding indices in the lists are greater than or less than each other. Yes! This is how Python does it. Huzzah! This does mean the elements at each index in each list must be the same type, though.

   D. List comparison using > or < is not allowed in Python. clearly false!

2B) also, no! if so then [10] would be less than [1] (because they match in their first character but 1 has less characters total). nonsense!

7) This question is oddly specific! I know I made it, but wow. LOL. I don't think you should expect something like this on the exam. This is mostly just memory/vocabulary, though. So not many notes to make here~

But it's neat to know the difference between .find() and .index()~

7. What is the difference between the `.find()` and `.index()` methods in Python?

   A. `.find()` works only on strings and returns -1 if the specified element is not found, while `.index()` works on strings and lists but raises an error if the specified element is not found.

   B. `.find()` only works on lists, and `.index()` only works on strings, but both return -1 if the specified element is not found.

   C. `.find()` and `.index()` are interchangeable methods, and there is no significant difference between them.

   D. `.find()` works on strings and lists, while `.index()` works only on strings, but both raise an error if the specified element is not found.

8. Which of the following statements accurately describes the uses of `list`()?

   A. `list()` is used to create an empty list. Yes! lst1 = list(). lst1 will be equal to []

   B. `list()` can convert a string into a list, where each character becomes an element Yes! rihanna = list("sos"). rihanna will be equal to ["s", "o", "s"]

   C. `list()` can convert other iterable objects like a range (from `range()`) into lists. Yes! list(range(3)) gets us [0, 1, 2]. But in the future, we see we can change sets and tuples to lists as well.

   D. All of the above. Yes!!!!

2	3	4	5	6	7	8
D	F	B	A	C	A	D

Page total: ________/14

The following 9 questions require you to trace the behavior of some Python code and identify the output of that code. For each question, write the output for the code segment in the box. If there's an error, say so.

9. (3 points)

```
school = ["nemo", "marlin", "dory"]
school2 = school
school2.append("gill")
print(school)
```

Note that school2 will refer to the same object as school, because lists are mutable. When you assign a list (or any mutable object) to another variable, you are actually creating a reference to the original object, not a new copy of it. So, by modifying school2, we are also modifying school.

["nemo", "marlin", "dory", "gill"]

10. (3 points)

```
doubleTrouble = ["garfield", "odie", "tom", "jerry"]
doubleTrouble = doubleTrouble.sort()
print(doubleTrouble)
```

Remember that .sort() modifies the list in-place (directly). It doesn't return a new list — it actually returns None.

None

11. (3 points)

This will be an error, because we're trying to add a list (toontown) with strings (clubhouse[i]).

```
toontown = []
clubhouse = ["mickey", "minnie", "pluto", "donald", "daisy", "goofy"]
for i in range(len(clubhouse)):
    toontown += clubhouse[i]
print(toontown)
```

For this to work, we need to put clubhouse[i] in a list ([clubhouse[i]]), so that we can add list to list.

Error

12. (3 points)

```
pink = ["courage", "cheshireCat", "pinkPanther"]
myFavoriteColor = ["courage", "cheshireCat", "pinkPanther"]
print(myFavoriteColor is pink and pink == myFavoriteColor)
```

False

pink and myFavoriteColor have the same items in them, so 'pink == myFavoriteColor' will be True. But because they are separate objects in memory, 'myFavoriteColor is pink' will evaluate to False. True and False will evaluate to False.

Page total: ________/12

Pink actually is my favorite color though…

13. (3 points)

```
answer = ["deepThought", "life", "universe", "everything", 42]
answer.sort()
print(answer)
```

This will be an error, because we're trying to sort a list that contains items of different types. We can't compare strings to an integer (42).

Error

14. (3 points)

```
myNums = [1, 2, 3, 4, 5]
print([num ** 2 for num in myNums])
```

OMG! An easy-ish question! Using list comprehension, we simply create a list where all elements in myNums are squared.

[1, 4, 9, 16, 25]

15. (3 points)

OK. I wanted this question to make kids think, if the loop lasts for as long as super is, but we append to super, does our loop have extra iterations?.

```
super = ["peach"]
shrooms = ["toad", "mario", "yoshi"]
for minion in range(len(super)):
    super.append(shrooms[minion])
print(super)
```

The answer is no. =) The length of super is obtained at the start of the loop (so, 1). So this loop will only have one iteration, so we add "toad" to super and that's it~

['peach', 'toad']

16. (3 points)

max(lst1, lst2) will compare items in each list. We compare "Bulbasaur" to "Squirtle". "Bulbasaur" is less than "Squirtle" in lexicographic order, so lst2 is less than lst1.

```
lst1 = ["Squirtle", "Snorlax"]; lst2 = ["Bulbasaur", "Charmander"]
print(min(max(lst1, lst2)))
```

We then do min(lst1), which compares "Squirtle" and "Snorlax". "Snorlax" is less than "Squirtle" lexicographically, so Snorlax is our final answer.

"Snorlax"

17. (3 points)

```
def iChooseYou(pc):
    pc.append("Pikachu")
    pc.sort()
pokemon = ["Onix", "Psyduck"]
iChooseYou(pokemon)
print(pokemon)
```

iChooseYou(pokemon) will append "Pikachu" to our list, and then it will sort the list. In lexicographic order (think the order we would encounter these words in a dictionary), "Onix" comes first, then "Pikachu", then "Psyduck".

These changes remain, even after/outside the function, because lists are mutable, so we are modifying this object directly.

['Onix', 'Pikachu', 'Psyduck']

Page total: ________/15