**Quiz 8**

To get credit for this quiz, use the Quiz tool at eee.uci.edu to enter your answers, within the Sunday-to-Tuesday quiz period.

There is a copy of the original ICStunes music manager program on the web at:

http://www.ics.uci.edu/~kay/python/ICStunes0.py

Open this file in a separate window; you will need to refer to it for this quiz.

**Problem 1** (4 points)  **Topic: Identifying data types**

Fill in each blank below with one data type from this group:

int float bool str list namedtuple tuple set dict Song Album Songdisplay

What data structure does the ICStunes code use to represent:

... the music collection? a ____________ of ____________ a list of album

... a single track on an album? a ____________ called ____________ a namedtuple called Song

... a single song along with info about that song's album? a ____________ called ____________

... the collection of songs on an album? a ____________ of ____________ a list of Song

**Problem 2** (5 points)  **Topic: Function as key parameter to sort() method**

Write the necessary code to sort the collection MUSIC into alphabetical order by the album artist's name. Following the existing code, this should consist of one statement plus one short function definition.

```python
def Album_artist(a: Album) -> str:
    ''' Return the artist field of the album
    '''
    return a.artist

MUSIC.sort(key=Album_artist)
```

**Problem 3** (9 points)  **Topic: Recognizing and incorporating previously defined functions**

(a) (5 points) Write the function `Album_average_length` that takes an album and returns the average length in seconds of a song on that album (as a float). If any of the functions already defined in the file are useful, you should use them for full credit.

```python
def Album_average_length(a: Album) -> float:
    ''' Return avg length in secs of a song on the album'''
    if len(a.songs) == 0:
        return 0
    else:
        return Album_length(a) / len(a.songs)
```

(b) (4 points) Write the one statement that will sort the collection MUSIC in order by the average length of each album's songs, greatest average first. `MUSIC.sort(key=Album_average_length, reverse=True)`
Problem 4 (3 points)  Topic: Identifying the right type of function as argument to sort()

The function `top_n_played` (the last definition in the file) uses `play_count_from_songdisplay` as the key argument to the `sort` method. Why doesn't it use `Song_play_count` instead?

Because `top_n_played` sorts a list of Songdisplays. If we're sorting a list of Songdisplays, the key function we use has to take a Songdisplay as its argument. (Recall that the function that's the value of the key argument to `sort()` takes one of the objects being sorted and returns a value based on that object, which is used for comparisons during the sort). `play_count_from_songdisplay` does take a Songdisplay, while `Song_play_count` takes a Song.

Problem 5 (6 points)  Topic: Familiarity with ICStunes code (given code base for reference)

In the code below, fill in each blank with one identifier, constant, or operator, consistent with the function header and docstring.

```python
def collection_search(C: [Album], search_for: str) -> [Songdisplay]:
    ''' Return a list of songdisplays that include (in the album title, artist, or song title) the specified string
    '''
    SDL = ____________(C)  all_Songdisplays
    result = [ ]
    for sd in SDL:
        if (__________ in sd.__________ or _________ in sd.__________ or _________ in sd.__________):
            result.___________(__________)
        search_for  a_title
        search_for  artist
        search_for  s_title
        append  sd
    return result
```

Problem 6 (3 points)  Topic: Categorizing operations as mapping, filtering, or reducing

Lab Assignment 8 described three categories of operations: mapping, filtering, and reducing. Below are three tasks on a collection of albums in the ICStunes music manager; identify which is a mapping operation, which is filtering, and which is reducing. There's exactly one of each.

(a) From a list of numbers representing the play-counts of each song in the collection, produce the total number of plays for the entire collection. **Reducing.**

(b) From a collection of albums, produce a list of strings, each string the title of an album. **Mapping.**

(c) From a collection of albums, produce a list of the albums released before the year 2000. **Filtering.**