Question 1. [7 MARKS]

The greatest video game music composer of all time is Motoi Sakuraba. Unfortunately, he has become mixed up with a bunch of other data in the Python dictionaries below. For each of the three subquestions on this page, add one or more lines of code to print `Sakuraba` **on a single line** by extracting the proper part of the dictionary. Each subquestion is independent.

Part (a) [1 MARK]

```
composers = {'a': 'Kondo', 'b': 'Sakuraba', 'c': 'Kikuta'}
```

```
print composers['b']
```

Part (b) [1 MARK]

```
composers = {'a': {'Sakuraba': 'b'}}
```

```
print composers['a'].keys()[0]
```

Part (c) [2 MARKS]

You are required to use a loop in this one.

```
composers = {1:'S', 2:'a', 3:'k', 4:'u',
             5:'r', 6:'a', 7:'b', 8:'a'}



for k in sorted(composers):
  print composers[k],
print
```

Part (d) [1 MARK]

Assume that I try to invert the following dictionary, using inversion functions we have written in lecture:

```
d = {'first': [1,2,3]}
```

Can this dictionary be inverted? Explain in one or two sentences what will happen.

No, the dictionary cannot be inverted. If you try, a key will be required to be a list, which is not allowed.

Part (e) [2 MARKS]

In the box beside the code below, write its output. If it would generate an error, say so, and give the reason for the error.

```
L1 = [[1, 2], [3, 4]]
L2 = L1[:]
L1[1].append (99)
print L1
print L2
```

```
[[1, 2], [3, 4, 99]]
[[1, 2], [3, 4, 99]]
```

Question 2. [6 MARKS]

A **composer** file consists of zero or more **composer blocks**. Each composer block consists of the following lines, in order:

- A line containing the composer's name
- Zero or more lines, each naming one song written by the composer

Each composer, except for the last, is followed by two lines of five asterisks each. Here is an example:

```
Uematsu
Silent Light
Sealed Door
*****
*****
Sasai
*****
*****
Kawasaki
Longing for the Past
Guardians
Unexplored Road
```

In this sample file, Uematsu has 2 songs; Sasai has 0 songs; and Kawasaki has 3 songs. Overall, there are three composers, five total songs; and the composer with the most number of songs is Kawasaki.

(See next page for question. You may use the area below for rough work, but it will not be marked unless you clearly indicate the part of it you want us to mark.)

Write the following function according to its docstring.

```
def composer_info (f):
  '''f is an open composer file. Return a list of three elements:  the first element is the number
  of composers in the file; the second is the total number of songs; and the third is the name
  of the composer (an str) with the most songs in the file (if multiple composers have the most
  songs, return any one of them).'''
```

Solution:

```
def composer_info (f):
  total_composers = 0
  total_songs = 0
  best_composer = ''
  best_composer_count = 0
  line = f.readline()
  while line:
    total_composers += 1
    composer_name = line
    song_counter = 0
    line = f.readline()
    while line and line != '*****\n':
      total_songs += 1
      song_counter += 1
      line = f.readline()
    f.readline()
    if song_counter > best_composer_count:
      best_composer_count = song_counter
      best_composer = composer_name
    line = f.readline()
  return [total_composers, total_songs, best_composer]
```

Question 3. [4 MARKS]

Write the following function according to its docstring.

```
def indices_and_elements (L):
  '''Return a list of tuples, one for each element in list L, in order. Each
  tuple should contain the index of the element from L, and the element
  itself. For example, indices_and_elements (['this', 'is', 'fun']) should
  return [(0, 'this'), (1, 'is'), (2, 'fun')]'''
```

Solution:

```
def indices_and_elements (lst):
  '''Return a list of tuples, one for each element in lst, in order. Each
  tuple should contain the index of the element from lst, and the element
  itself. For example, indices_and_elements (['this', 'is', 'fun']) should
  return [(0, 'this'), (1, 'is'), (2, 'fun')]'''

  tuples = []
  for i in range(len(lst)):
    tuples.append ((i, lst[i]))
  return tuples
```