Question 1. [6 MARKS]

Part (a) [1 MARK] What is the output of the following program?

```
num = 84
if num >= 80:
  print 'A'
if num >= 20:
  print 'B'
else:
  print 'C'
```

Solution:

```
A
B
```

Part (b) [1 MARK] What does this program print? (Assume that `absent.wav` exists.)

```
snd1 = sound.load_sound ('absent.wav')
snd2 = sound.load_sound ('absent.wav')
print id(snd1) == id(snd2)
```

Solution:

```
False
```

Part (c) [1 MARK] Assume that `s` refers to a string with at least two characters. Write an expression that evaluates to the last two characters in `s`.

Solution:

```
s[-2:]
```

Part (d) [1 MARK] Fill-in the missing expression so that the while-loop never executes.

```
response = ____________________________________________________________
while response == 'a' or response == 'b':
  response = raw_input ('Type something: ')
```

Solution:

```
response = 'c' # many answers
```

Part (e) [1 MARK] What is the output of the following program?

```
def blah(x):
  x = 1982

x = 2
blah(x)
print x
```

Solution:

```
2
```

Part (f) [1 MARK] Briefly explain the difference between a function definition and a function call.

Solution: A function definition tells Python what a function does: it specifies the function's name, the parameters, and the body. A function call runs the function's code using specific values for the parameters.

Question 2. [7 MARKS]

This question asks you to write a program in two steps. First, you'll write a function that determines whether each sample in a sound has a left value equal to its right value. Then, you'll write a main block that calls this function based on user inputs. Assume that `sound` has already been `import`ed.

Part (a) [4 MARKS]

On the next page, write the function according to its docstring. As an example, if you call `all_equal` with a sound whose three samples are (10, 10), (40, 40), and (-523, -523), then `True` should be returned. As a second example, if you call `all_equal` with a sound whose two samples are (5, 6) and (40, 40), then `False` should be returned.

```
def all_equal (snd):
  '''Return True if all samples in Sound snd
  have their left channel value equal to their right channel value.
  Return False otherwise.'''
```

Part (b) [3 MARKS]

Complete the main block below. Your program should first use `raw_input` to ask the user for the name of a wav file; use the prompt `Enter filename:`. Then, your program should output one of the following two strings, depending on the output of `all_equal`:

- If `all_equal` returns `True`, output `Sound is mono`
- If `all_equal` returns `False`, output `Sound is not mono`

```
if __name__ == '__main__':
```

Solution:

```
def all_equal (snd):
  '''Return True if all samples in Sound snd
  have their left channel value equal to their right channel value.
  Return False otherwise.

  for samp in snd:
    if sound.get_left(samp) != sound.get_right(samp):
      return False
  return True


if __name__ == '__main__':
  fname = raw_input ("Enter filename: ")
  snd = sound.load_sound (fname)
  res = all_equal (res)
  if res:
    print 'Sound is mono'
  else:
    print 'Sound is not mono'
```

Marking Scheme:

```
For part A:
  +1 for the for-loop
  +2 for an 'if' with a 'return False'
  +1 for 'return True' at the end

For part B:
  +1 for the raw_input
  +0.5 for loading the sound
  +0.5 for calling the function
  +1 for the 'if' and associated output messages
```

Question 3. [5 MARKS]

Write the following function according to its docstring. You **must** use a while-loop in your solution (if you don't, no `absent.wav` for you!). Use the prompt `Enter a string:` when prompting for a string.

For example, if I call the function as follows:

```
prefixed_strings(3, 'wh')
```

and then type the following three lines:

```
knock knock!
who's there?
no one. people don't visit anymore. they Skype! how didn't you know that?
```

the function would return `1` (because only one string starts with `wh`).

```
def prefixed_strings (num, prefix):
  '''num is a positive int; prefix is a string.
  Prompt the user for a total of num strings,
  and return the number of those strings that start with prefix.'''
```

Solution:

```
def prefixed_strings (num, prefix):
  good = 0
  counter = 0
  while counter < num:
    s = raw_input ('Enter a string: ')
    if s.startswith (prefix):
      good += 1
    counter += 1
  return good
```

Marking Scheme:

```
+1 for initializing two counters
+1 for the correct while guard
+0.5 for the raw_input
+1 for the correct 'startswith'
+1 for incrementing the good counter
+0.5 for returning the good counter
```