Do not turn this page until you have received the signal to start.
(Please fill out the identification section above, write your name on the back of the test, and read the instructions below.)

Good Luck!

This midterm consists of 3 questions on 8 pages (including this one). When you receive the signal to start, please make sure that your copy is complete. Comments are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can’t figure out how to write the code.
If you use any space for rough work, indicate clearly what you want marked.  

# 1: _____/ 6
# 2: _____/ 7
# 3: _____/ 5

TOTAL: _____/18
[Use the space below for rough work. This page will not be marked, unless you clearly indicate the part of your work that you want us to mark.]
Question 1. [6 MARKS]

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

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

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

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

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`.

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

```python
response = _________________________________
while response == 'a' or response == 'b':
    response = raw_input ('Type something: ')```
Part (e) [1 MARK] What is the output of the following program?

def blah(x):
    x = 1982

    x = 2
    blah(x)
    print x

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

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 imported.

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__':
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:

```python
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.'''
```

def 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).

```python
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.'''
```
Short Python function/method descriptions:

__builtins__:
- abs(number) -> number
  Return the absolute value of the given number.
- max(a, b, c, ...) -> value
  With two or more arguments, return the largest argument.
- min(a, b, c, ...) -> value
  With two or more arguments, return the smallest argument.
- raw_input([prompt]) -> str
  Read a string from standard input. The trailing newline is stripped. The prompt string, if given, is printed without a trailing newline before reading.

int:
- int(x) -> int
  Convert a string or number to an integer, if possible. A floating point argument will be truncated towards zero.

media:
- choose_file() -> str
  Prompt user to pick a file. Return the path to that file.
- create_sound(int) -> Sound
  Create a sound with the specified number of samples. All sample values are 0.
- get_left(sample) -> int
  Return the left value of the given sample.
- get_right(sample) -> int
  Return the right value of the given sample.
- load_sound(str) -> Sound
  Return a Sound object from file with the given filename.
- set_left(sample, int)
  Set the left value of the given sample to the given int value.
- set_right(sample, int)
  Set the right value of the given sample to the given int value.
- play(Sound)
  Play the given Sound.

str:
- x in s -> bool
  Return True if x is in s, and False otherwise.
- str(x) -> str
  Convert an object into its string representation, if possible.
- S.count(sub[, start[, end]]) -> int
  Return the number of non-overlapping occurrences of substring sub in string S[start:end]. Optional arguments start and end are interpreted as in slice notation.
- S.find(sub[,i]) -> int
  Return the lowest index in S (starting at S[i], if i is given) where the string sub is found or -1 if sub does not occur in S.
- S.isdigit() -> bool
  Return True if all characters in S are digits and False otherwise.
- S.lower() -> str
  Return a copy of the string S converted to lowercase.
- S.startswith (sub) -> bool
  Return True if s starts with substring sub, and False otherwise.
- S.strip() -> str
  Return a copy of S with leading and trailing whitespace removed.
- S.upper() -> str
  Return a copy of the string S converted to uppercase.