You must earn at least 30 out of 75 marks (40%) on this final examination in order to pass the course. Otherwise, your final course grade will be no higher than 47%.

Student Number: _______________

Last (Family) Name(s): ______________________________________

First (Given) Name(s): ______________________________________

Do not turn this page until you have received the signal to start. In the meantime, please read the instructions below carefully.

This Final Examination paper consists of 9 questions on 22 pages (including this one), printed on both sides of the paper. When you receive the signal to start, please make sure that your copy of the paper is complete and fill in your name and student number above.

- Comments and docstrings are not required except where indicated, although they may help us mark your answers.
- You do not need to put import statements in your answers.
- No error checking is required: assume all user input and all argument values are valid.
- Do not use break or continue on this exam.
- If you use any space for rough work, indicate clearly what you want marked.
- Do not remove pages or take the exam apart.

Marking Guide

# 1: _____/ 9
# 2: _____/10
# 3: _____/ 4
# 4: _____/ 8
# 5: _____/10
# 6: _____/10
# 7: _____/ 7
# 8: _____/ 8
# 9: _____/ 9

TOTAL: _____/75
Short Python function/method descriptions:

```python
__builtins__:
    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.
    abs(x) -> number
    Return the absolute value of x.
    chr(i) -> Unicode character
    Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.
    int(x) -> int
    Convert x to an integer, if possible. A floating point argument will be truncated
    towards zero.
    len(x) -> int
    Return the length of the list, tuple, dict, or string x.
    max( iterable ) -> object
    max(a, b, c, ...) -> object
    With a single iterable argument, return its largest item.
    With two or more arguments, return the largest argument.
    min( iterable ) -> object
    min(a, b, c, ...) -> object
    With a single iterable argument, return its smallest item.
    With two or more arguments, return the smallest argument.
    open( name[, mode]) -> file open for reading, writing, or appending
    Open a file. Legal modes are "r" (read), "w" (write), and "a" (append).
    ord(c) -> integer
    Return the integer ordinal of a one-character string.
    print( value,..., sep=' ', end='
') -> NoneType
    Prints the values. Optional keyword arguments:
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    range(start, stop, [step]) -> list-like-object of int
    Return the integers starting with start and ending with stop - 1 with step specifying
    the amount to increment (or decrement).
    If start is not specified, the list starts at 0. If step is not specified,
    the values are incremented by 1.

dict:
    D[k] -> object
    Produce the value associated with the key k in D.
    del D[k]
    Remove D[k] from D.
    k in d -> bool
    Produce True if k is a key in D and False otherwise.
    D.get(k) -> object
    Return D[k] if k in D, otherwise return None.
    D.keys() -> list-like-object of object
    Return the keys of D.
    D.values() -> list-like-object of object
    Return the values associated with the keys of D.
    D.items() -> list-like-object of tuple of (object, object)
    Return the (key, value) pairs of D, as 2-tuples.
```
file open for reading:
F.close() -> NoneType
   Close the file.
F.read() -> str
   Read until EOF (End Of File) is reached, and return as a string.
F.readline() -> str
   Read and return the next line from the file, as a string. Retain any newline.
F.readlines() -> list of str
   Return a list of the lines from the file. Each string retains any newline.

file open for writing:
F.close() -> NoneType
   Close the file.
F.write(x) -> int
   Write the string x to file F and return the number of characters written.

list:
x in L -> bool
   Produce True if x is in L and False otherwise.
L.append(x) -> NoneType
   Append x to the end of the list L.
L.extend( iterable) -> NoneType
   Extend list L by appending elements from the iterable. Strings and lists are iterables whose elements are characters and list items respectively.
L.index( value) -> int
   Return the lowest index of value in L.
L.insert( index, x) -> NoneType
   Insert x at position index.
L.pop( [index]) -> object
   Remove and return item at index (default last).
L.remove( value) -> NoneType
   Remove the first occurrence of value from L.
L.reverse() -> NoneType
   Reverse *IN PLACE*.
L.sort() -> NoneType
   Sort the list in ascending order *IN PLACE*.

str:
x in s -> bool
   Produce True if and only if x is in s.
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.endswith(S2) -> bool
   Return True if and only if S ends with S2.
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.index( sub) -> int
   Like find but raises an exception if sub does not occur in S.
S.isalpha() -> bool
   Return True if and only if all characters in S are alphabetic
   and there is at least one character in S.
S.isdigit() -> bool
   Return True if all characters in S are digits
   and there is at least one character in S, and False otherwise.
S.islower() -> bool
   Return True if and only if all cased characters in S are lowercase
   and there is at least one cased character in S.
S.isupper() -> bool
   Return True if and only if all cased characters in S are uppercase
   and there is at least one cased character in S.
S.lower() -> str
   Return a copy of the string S converted to lowercase.
S.lstrip([chars]) -> str
   Return a copy of the string S with leading whitespace removed.
   If chars is given and not None, remove characters in chars instead.
S.replace(old, new) -> str
   Return a copy of string S with all occurrences of the string old replaced
   with the string new.
S.rstrip([chars]) -> str
   Return a copy of the string S with trailing whitespace removed.
   If chars is given and not None, remove characters in chars instead.
S.split([sep]) -> list of str
   Return a list of the words in S, using string sep as the separator and
   any whitespace string if sep is not specified.
S.startswith(S2) -> bool
   Return True if and only if S starts with S2.
S.strip([chars]) -> str
   Return a copy of S with leading and trailing whitespace removed.
   If chars is given and not None, remove characters in chars instead.
S.upper() -> str
   Return a copy of the string S converted to uppercase.

random:
shuffle(L) -> NoneType
   Randomly shuffle (re-order) list L in place, and return None.