1. Consider this code

```python
wish = 'Happy Birthday'
```

Assuming the code above has been executed, circle the expression(s) that produce 'happy birthday'.

(a) `wish[0].lower() + wish[6].lower()`  
(b) `wish.swapcase()`  
(c) `wish[0].lower() + wish[1:6] + wish[6].lower() + wish[7:]`  
(d) `wish.lower()`

2. Consider this code

```python
robot = 'R2D2'
```

Assuming the code above has been executed, circle the expression(s) that produce True.

(a) `robot.isupper()`  
(b) `robot.isalpha()`  
(c) `robot.isalnum()`  
(d) `robot.isdigit()`

3. Consider this code

```python
lyrics = '''O Canada!  
Our home and native land!  
True patriot love in all thy sons command.'''
```

Circle the expression that produces the index of the second exclamation mark.

(a) `lyrics.find('!!')`  
(b) `lyrics.find('!!').find('!!')`  
(c) `lyrics.find('!!', lyrics.find('!!'))`  
(d) `lyrics.find('!!', lyrics.find('!!')) + 1`
Short Python help descriptions:

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.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.isalnum() --> bool
  Return True if and only if all characters in S are alphanumeric
  and there is at least one character 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 and only if all characters in S are digits
  and there is at least one character in S.
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.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.swapcase() --> str
  Return a copy of S with uppercase characters converted to lowercase
  and vice versa.
S.upper() --> str
  Return a copy of the string S converted to uppercase.