Midterm solutions and results

There were two versions of the test, with essentially identical questions, just dressed up differently. Both exam versions and the solutions are appended below.

A few comments on the questions and the solutions: A somewhat common error in Problem 1 was to say that 1(b) was something different from a boolean, and that 1(a) was a float rather than an int.

It was common to try to solve the problem in 2(a) at the level of the function rather than the main program. I gave 4/6 for that—this answer was not consistent with the answer given in the last part of the problem, so that it did not get full credit. The reason is that if you convert to a float or an int in the function, then the arguments to the function have to be considered as strings, rather than ints or floats, and hardly any student wrote 2(d) this way.

The solution for the first part of 2(b) is just an example—any combination of values that leads to the same error is okay.

There was an error in question 2(d) in the sense that the parameter names should have been chosen to refer to GPA, rather than years (it’s pretty obvious why this happened!), but this did not seem to cause a problem for anyone.

The most common incorrect answer was 4(b): The list method sort (like append, but unlike count) does not return a value. Originally I took off all 6 points for this, then relented and gave half credit if you gave the sorted list as a value. Because of many minor adjustments like this, made in an effort to grade consistently and accurately, there are a lot of cross-outs in the score on your paper.)

How’d you do?

42 students took the exam. More than half the class received scores of 80 or above, which is very good. If your score was in this range, then there are no real concerns; you may have misunderstood a fine point. If your score was in the 70-79 range, look closely at what you got wrong—no serious concerns, but you might be missing something important.

A score below 60, and especially a score below 50, is a cause for real concern. If you are in this category, please arrange to see me so we can discuss your progress in the course.
There are 16 short-answer questions, each worth 6 points, and 4 points will be added to your score. Total: $16 \times 6 + 4 = 100$ points.

Write your name on the BACK of each of the two answer sheets. (This lets me grade anonymously.)

Write all of your answers on the answer sheet provided on the last two pages of the exam. You may use the other pages for scratch work.

You can bring in one sheet of notes, front and back.

Read over every question; answer the ones that you know how to answer easily first, then go back to questions that you have to spend more time on.
1. **Value and Type.** Each part of this problem gives a Python expression---in one case the expression comes at the end of a sequence of statements. In each case you are to give both the value and the type of the expression. The type will always be `int`, `float`, `str`, `bool`, `list` or `NoneType`. All the statements here are legal and can be executed without causing a run-time error.

(a) $5+6/3$

(b) ‘L’ in ‘Boston College’.upper()

(c) ‘Boston College’[2:5][2]

(d) a=['broccoli','asparagus','cabbage']
   a.sort()
   a[1]

(e) [3*x+1 for x in range(2,5)]
2. **Conditionals.** To be eligible for retirement benefits, a company’s employee must (a) be at least 60 years of age and have worked for the company for at least 25 years, OR (b) be at least 65 years old OR (c) have worked for the company for at least 30 years. The program below is designed to allow employees to enter their personal information and find if they are eligible for the benefits.

```python
def eligible(age, years):
    if age >= 60:
        if years >= 25:
            return True
        else:
            return False
    elif age >= 65:
        return True
    elif years >= 30:
        return True
    return False

y = input('Enter years in service. ')
a = input('Enter age. ')
if eligible(a, y):
    print('You are eligible for retirement benefits. ')
else:
    print('Sorry, you will have to keep working. ')
```

(a) When the program was first tested, it crashed immediately, and printed the error message:

```
if age >= 60:
    TypeError: '>=' not supported between instances of 'str' and 'int'
```

Tell as simply as possible (one sentence) how the code should be changed to fix this error.

(b) Once that repair was made, the program ran smoothly. But it was eventually discovered that it was giving incorrect results. Describe a situation (age and number of years in service) in which an employee eligible for retirement is denied benefits, and tell *as simply as possible* how to fix it. (This will not require extensive rewriting of code---a change to one or two lines will be sufficient.)

(c) The function `eligible`, (the correct version) can be written much more concisely, so that its body consists of just *a single line of code*, with no occurrence of *if*. Write this line of code. (If you cannot do it in one line, do it with as small a number of lines of code as you are able.)
3 A nested loop. The program below is meant to print out the following gorgeous design:

```
  a b c d e f g
  b c d e f g
  c d e f g
  d e f g
  e f g
  f g
  g
```

```python
letters='abcdefg'
row=0
while row<7:
    for j in range(7-row):
        print(letters[j],end='')
print()
```

The code contains 3 errors. You will describe how to fix each of the errors in turn. In each case the correction involves either inserting or moving a very small amount of code, never more than a line. If you insert or move a line of code, make sure you tell its indentation level---there are three different levels of indentation in the code above. (By default, the print function advances to the next line after printing its arguments, so that the next thing that is printed will appear on a different line. The argument end='' in the print function changes this, so that the next thing that is printed will begin at the very next character.)

(a) First error: When the program is first run, the user’s screen fills up with letters, and the program has to be forced to quit.
(b) Second error: After this is fixed, the program prints a single line of letters, then a blank line, and then stops.
(c) Third error: After these are fixed, the pattern of letters that is printed out has the right shape, but the wrong letters.
(d) After all the errors are corrected, the programmer realizes that she could have replaced the last 5 lines of code with two very compact lines: a for statement that has just a single print statement in its body. Write this for statement.
4 Modifying a list. This problem concerns the function below.

```python
def listFunction(L):
    newlist=[]
    for x in L:
        if x<=len(L):
            newlist.append(x)
    return newlist
```

(a) What is the value of the expression

```
listFunction ([2,7,-3,8,4,9])
```

(b) What would be the result if the last line of the function were changed to

```
return newlist.sort()
```

(c) What is the value of the expression

```
listFunction(""")
```

(The argument is the empty string).

(d) Write an equivalent version of `listFunction` that only has a single line, using list comprehension.
1. (a) Type \texttt{float} \hspace{1cm} \text{Value: 7.0} \\
(b) Type: \texttt{bool} \hspace{1cm} \text{Value: True} \\
(c) Type: \texttt{str} \hspace{1cm} \text{Value: ‘o’} \\
(d) Type: \texttt{str} \hspace{1cm} \text{Value: ‘broccoli’} \\
(e) Type: \texttt{list} \hspace{1cm} \text{Value: [7,10,13]} \\
(f) Type: \text{There was no part (f)!} \\

2. (a) Modification that prevents the crash: 
\text{Apply \texttt{int()} or \texttt{float()} to the result of input before calling the function.} \\
(b) Parameter values that will lead to incorrect result. 
\text{Age: 65 \hspace{1cm} Years in service: 24} \\
\text{Modification to prevent this error:} \\
\text{Get rid of else: \texttt{return False} in lines 5-6 of function definition} \\
(c) The shorter version of the function: 
\begin{verbatim}
def eligible(age, years):
    return (age>=60 and years>=25) or (age>=60) or years>=30
\end{verbatim}
3 (a) Modification that corrects first error (program does not stop):

row+=1 should be last statement within while loop (indent level 1, even with the word for).

(b) Modification that corrects second error (single line of output):
print() should be moved within body of while statement (again, indent level 1)

(c) Modification that corrects third error (right pattern, wrong letters):
letters[j] should be replaced by letters[j+row]

(d) Concise version of program:

```python
letters = 'abcdefg'

for row in range(7):
    print(letters[row:]
```

4. (a) [2,-3,4]

(b) None

(c) [] (empty list)

(d) Concise version of function:

```python
def listFunction(L):
    return [x for x in L if x <= len(L)]
```
There are 16 short-answer questions, each worth 6 points, and 4 points will be added to your score. Total: $16 \times 6 + 4 = 100$ points.

Write your name on the BACK of each of the two answer sheets. (This lets me grade anonymously.)

Write all of your answers on the answer sheet provided on the last two pages of the exam. You may use the other pages for scratch work.

You can bring in one sheet of notes, front and back.

Read over every question; answer the ones that you know how to answer easily first, then go back to questions that you have to spend more time on.
3. **Value and Type.** Each part of this problem gives a Python expression---in one case the expression comes at the end of a sequence of statements. In each case you are to give both the value and the type of the expression. The type will always be int, float, str, bool, list or NoneType. There are no errors---every sequence of statements here is legal.

(f) $6+12/4$

(g) ‘C’ in ‘Boston College’.lower()

(h) ‘Boston College’[3:6][1]

(i) 
```python
a=['Washington','Adams','Jefferson','Madison']
a.sort()
a[3]
```

(j) 
```python
[4*x-1 for x in range(2,5)]
```
4. **Conditionals.** To be eligible for a certain scholarship, a student must either (a) maintain a cumulative GPA for all enrolled semesters of 3.2 and have no grade in the current semester lower than 2.67, OR (b) maintain a cumulative GPA for all enrolled semesters of at least 3.4, OR (c) have no grade in the current semester lower than 3.0. The program below asks a student to enter in both the cumulative GPA and the lowest current grade, and determines eligibility.

```python
def eligible(cum, low):
    if cum >= 3.2:
        if low >= 2.67:
            return True
        else:
            return False
    elif cum >= 3.4:
        return True
    elif low >= 3.0:
        return True
    return False

cumulative = input('Enter GPA over all semesters')
lowest = input('Enter lowest current grade as GPA')
if eligible(cumulative, lowest):
    print('Eligible.')</nelse:
    print('Ineligible.')</n```

(a) When the program was first tested, it crashed immediately, and printed the error message:

```
if cum >= 3.2:
TypeError: '>= ' not supported between instances of 'str' and 'float'
```

Tell as simply as possible (one sentence) how the code should be changed to fix this error.

(b) Once that repair was made, the program ran smoothly. But it was eventually discovered that it was giving incorrect results. Describe a situation (cumulative GPA and lowest current grade) in which a qualifying student is denied a scholarship, and tell as simply as possible how to fix it. (This will not require extensive rewriting of code---a change to one or two lines will be sufficient.)

(c) The function `eligible` (the correct version) can be written much more concisely, so that its body consists of just a single line of code, with no occurrence of `if`. Write this line of code. (If you cannot do it in one line, do it with as small a number of lines of code as you are able.)
5 **A nested loop.** The program below is meant to print out the following gorgeous design:

```
letters='ABCDEFGH'
row=0
while row<8:
    for j in range(8-row):
        print(letters[j],end=' ')
print()
```

The code contains 3 errors. You will describe how to fix each of the errors in turn. In each case the correction involves either inserting or moving a very small amount of code, never more than a line. If you insert or move a line of code, make sure you tell its indentation level---there are three different levels of indentation in the code above. (By default, the print function advances to the next line after printing its arguments, so that the next thing that is printed will appear on a different line. The argument end="" in the print function changes this, so that the next thing that is printed will begin at the very next character.)

(e) *First error:* When the program is first run, the user’s screen fills up with letters, and the program has to be forced to quit.

(f) *Second error:* After this is fixed, the program prints a single line of letters, then a blank line, and then stops.

(g) *Third error:* After these are fixed, the pattern of letters that is printed out has the right shape, but the wrong letters.

(h) After all the errors are corrected, the programmer realizes that she could have replaced the last 5 lines of code with two very compact lines: a `for` statement that has just a single print statement in its body. Write this `for` statement.
6 **Modifying a list.** This problem concerns the function below.

```python
def listFunction(L):
    newlist=[]
    for x in L:
        if x>len(L):
            newlist.append(x)
    return newlist
```

(f) What is the value of the expression 

```python
listFunction([2,7,-3,8,4,9])
```

(g) What would be the result if the last line of the function were changed to 

```python
return newlist.sort()
```

(h) What is the value of the expression 

```python
listFunction([0])
```

(i) Write an equivalent version of `listFunction` that only has a single line, using list comprehension.
3. (a) Type: `float` Value: 9.0
   (b) Type: `bool` Value: `False`
   (c) Type: `str` Value: ‘o’
   (d) Type: `str` Value: ‘Washington’
   (j) Type: `list` Value: `[7,11,15]`
   (f) There is no part (f)

4. (a) Modification that prevents the crash:
   apply the `float()` function to the strings returned by input before calling `eligible` function

   (b) Parameter values that will lead to incorrect result.
   Cumulative GPA: 3.4    Lowest grade in current semester: 2.33

   Modification to prevent this error:
   Eliminate `else: return False` in lines 5-6 of function definition

   (c) The shorter version of the function: (Note that parameter names should be `cum` and `low`—this was a typo in the exam.)

   ```python
def eligible(cum, low):
    return (cum>=3.2 and low>=2.67) or (cum>=3.4) or (low>=3.0)
```
3 (a) Modification that corrects first error (necessary to force quit):

Add `row+=1` as last statement within while loop (indent level 1, even with the word for)

(b) Modification that corrects second error (single line of output):
Move print() statement to the same indent level as above (within while, not within for)

(c) Modification that corrects third error (right pattern, wrong letters):
Change `letters[j]` to `letters[j+row]`

(d) Concise version of program:

```python
letters = 'ABCDEFGH'
for row in range(8):
    print(letters[row:row+1])
```

4.(a) `[7,8,9]`

(b) None

(c) `[]` empty list

(d) Concise version of function:

```python
def listFunction(L):
    return [x for x in L if x>len(L)]
```