ICS 31 • UC Irvine
Fall 2014 • David G. Kay

Your Name _____ Your lab:
Your Student ID (8 digits) _____ Section (1-16) _____
Your UCInet ID _____ Time MWF at: 8a 10 12 2 4 6 8p
TA's name _____

First Midterm

You have 75 minutes (until the end of the class period) to complete this exam. There are 60 points possible, so allow approximately one minute per point and you'll have plenty of time left over.

Please read all the problems carefully. If you have a question on what a problem means or what it calls for, ask us. Unless a problem specifically asks about errors, you should assume that each problem is correct and solvable; ask us if you believe otherwise.

In answering these questions, you may use any Python 3 features we have covered in class, in the text, in the lab assignments, or earlier on the exam, unless a problem says otherwise. Use more advanced features at your own risk; you must use them correctly. If a question asks for a single item (e.g., one word, identifier, or constant), supplying more than one will probably not receive credit.

Remember, stay cool! If you run into trouble on a problem, go on to the next one. Later on, you can go back if you have time. Don't let yourself get stuck on any one problem.

You may not share with or receive from anyone besides the instructor or TAs any information or materials during the exam. You may not use any electronic devices.

Please write your answers clearly and neatly—we can't give you credit if we can't decipher what you've written.

We'll give partial credit for partially correct answers, so writing something is better than writing nothing. But be sure to answer just what the question asks.

Good luck!

Problem 1 (13 points)
Problem 2 (9 points)
Problem 3 (15 points)
Problem 4 (12 points)
Problem 5 (8 points)
Problem 6 (3 points)
Total (60 points)

Problem 1 (13 points)

Use the following definitions in this problem:

```
m = 'October'
h = 'Halloween'
d = 31
characters = ['pumpkin', 'ghost', 'skeleton', 'witch', 'black cat', 'spider']
```

(a) (2 points) What does Python print as it executes the following sequence of statements? (Write your answers in the blank space to the right of the code.)

```
print("Time to start!")
print("Don't be scared.")
print(m, d)
print(len(characters))
print(13 + (d * 100))
```

(b) (8 points) What does Python print as it executes the following sequence of statements? (Write your answers in the blank space to the right of the code.) Remember zero-based indexing.

```
print(characters[2])
print(characters[3], 'hat')

print(len(h) > 5)
print(h[0], m[0])

print(h[-1] == m[-1])
print('cat' in characters, 'low' in h)

print(characters[-1], 'web')
print(m[0:3], (d+1)/2)
```

(c) (3 points) What does the following code print out?

```
print("Boo!")
if d < 100:
    print(m)
    print(d)
else:
    print(h)
print("Trick or treat!")
```

Problem 2 (9 points)

Anteater Kitchen Supply sells kitchen equipment. They represent each item they sell in a namedtuple called Product that has four fields: the name of the item, the manufacturer of the item, the cost of the item from the manufacturer to AKS, and the price of the item to AKS's customers.

(a) (3 points) Which of the following defines a namedtuple that satisfies this specification? Circle *one or more* of A, B, C, D, E, or F; more than one may be correct.

A. `Product = namedtuple('Product', 'name manufacturer cost price')`
B. `Product = namedtuple('Chef Knife', 'Ginsu Knives', 10.00, 24.99)`
C. `Product = namedtuple('Product', 'name; manufacturer; cost; price')`
D. `Product = namedtuple('Product', 'item_name mfg wholesale_cost retail_price')`
E. `Product = namedtuple('Product', 'name manufacturer price')`
F. `Product = Product('Bread Knife', 'Henckels', 40.00, 62.50)`

(b) (3 points) Which of the following creates a new Product object, following the definition above, to represent a mixing bowl made by Corning that sells to customers for $9.99 and costs AKS $4.50? Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `new_product = Product.mixing_bowl('Corning', 4.50, 9.99)`
B. `new_product = Product('Mixing Bowl, Corning, 4.50, 9.99')`
C. `new_product = Product('Mixing Bowl', 'Corning', 4.50, 9.99)`
D. `new_product = namedtuple('Product', 'Mixing Bowl', 'Corning', 4.50, 9.99)`
E. `new_product = Product(9.99, 'Mixing Bowl', 'Corning', 4.50)`

(c) (3 points) The profit on each product AKS sells is the difference between its cost from the manufacturer and what AKS charges its customer. We write the `profit()` function below to compute the profit on selling a specified number of a given product.

```
def profit(P: Product, quantity: int) -> float:
    """ Return the profit on selling the specified quantity of the given Product. """
    return _____________________________________

assert profit(new_product, 1) == 5.49
assert profit(new_product, 10) == 54.90
```

Which of the following could we insert as the return value to implement `profit()` correctly (according to at least one of the correct definitions in part (a))? Circle *one or more* of A, B, C, D, E, or F; more than one may be correct.

A. `(P.price - P.cost) * quantity`
B. `P._replace(profit = quantity * (price - cost))`
C. `quantity * P.price - P.cost`
D. `Product(P.name, P.manufacturer, P.cost * quantity, P.price * quantity)`
E. `print((P.price - P.cost) * quantity)`
F. `quantity * (P.price - P.cost)`

Problem 3 (15 points)

Professor Priscilla Programmer teaches an advanced software engineering course in which the final programming project is graded on three criteria: correctness, efficiency, and style. She represents each student with a namedtuple defined as follows:

```
Student = namedtuple('Student', 'name ID correctness efficiency style')
```

The name is a string, the ID is an int, and the remaining fields are floats storing the scores on each criterion (in the range 0 to 100).

(a) (2 points) In the function definition below, fill in each blank with a single Python constant, operator, or identifier name (variable, function, attribute, method) to satisfy the problem specification.

```
def project_score(s: Student) -> float:
    """ Return the student's score on the final programming project, with correctness
        worth 75%, efficiency worth 10%, and style worth 15%
    """
    return s.correctness * ________ + s.________ * 0.10 ________ s.style ________ 0.15
```

(b) (5 points) What do the following statements print (assuming a correct solution to part (a))? The arithmetic is doable in your head.

```
S1 = Student('Anteater, Anthony', 33445566, 100, 100, 100)
S2 = Student('Zotter, Zelda', 44556677, 50, 50, 50)
S3 = Student('Irvine, Ervin', 55667788, 100, 0, 0)
print(S1.name, project_score(S1))
print(S2.name, project_score(S2))
print(S3.name, project_score(S3))
```

(c) (4 points) If Prof. Programmer has a list of 700 students called `SL`, fill in the blanks below to reorder `SL` by each student's overall final project score, highest to lowest:

```
_________________.sort(key=_________________, reverse=_________________)
```

(d) (4 points) In the function definition below, fill in each blank with a single Python constant, operator, or identifier name (variable, function, attribute, method) to satisfy the problem specification.

```
def star_students (LS: 'list of Student') -> 'list of Student':
    """ Return a list of those students on the input list whose project score is
        95 or greater.
    """
    result = [ ]
    for s in _____________:
        if _____________(s) >= 95:
            _____________.append(s)
    return _____________
```

Problem 4 (12 points)

Suppose we have a list of Students called `SL`, as in the previous problem. Fill in each blank with one of the ten code segments below (A through J) to match its most accurate description. Don't use any segment (A–J) more than once.

A.
```
for s in SL:
    print(s)
```

B.
```
for s in sorted(SL, key=project_score):
    print(s.name, project_score(s) > s.correctness)
```

C.
```
L = sorted(SL, key=project_score, reverse=True):
print(L[0:4])
```

D.
```
for s in sorted(SL, key=project_score):
    if project_score(s) > s.correctness:
        print(s.name, s.project_score)
```

E.
```
for s in SL:
    print(s.name, project_score(s))
```

F.
```
for s in sorted(SL, key=project_score):
    if project_score(s) > s.correctness:
        print(s.name)
```

G.
```
for s in SL:
    if s.correctness < 75:
        print(s.name)
```

H.
```
L = sorted(SL, key=project_score, reverse=True):
for i in range(5):
    print(L[i].name)
```

I.
```
for s in SL:
    print(SL[s].name, project_score(SL[s]))
```

J.
```
L = sorted(SL, key=project_score, reverse=True):
for s in L:
    if project_score(s) < 75:
        print(s.name)
```

___ Produces an error message about Student not having a project_score attribute

___ Prints everything stored about each Student in Python namedtuple form

___ Prints the names of the five top-scoring students

___ Prints the name of each student with his or her overall final project score

___ Prints the names of the students whose correctness score is under 75

___ Prints the names of the students whose overall final project score is greater than their correctness score

Problem 5 (8 points)

In this problem use definitions that appear elsewhere in this exam where appropriate. Choose your answers from these data types:

`int` `float` `bool` `str` `Product` `Student` list of `Product` list of `Student` list of `str` function

(a) (4 points) In each of the following Python expressions or statements, indicate what *data type* belongs in the indicated place.

```
line = _____________ + '\n'
print(_____________.price)
if _____________:
SL[_____________] = S2
```

```
bonus = _____________.style + 10
print(_____________[0].style)
PL.sort(key=_____________)
_____________ = 3.7 / x - a
```

(b) (4 points) Identify the *data type* of each of the following expressions.

```
len(characters[1])
profit
project_score(S2)
S1.efficiency >= S2.efficiency
```

```
characters
SL[-1]
SL
sorted(SL)
```

Problem 6 (3 points)

Each of the following is a good programming practice we've discussed in class, *except one.* Which of the following is *not* a good programming practice we discussed? Circle one of A, B, C, D, or E.

A. Avoiding duplicate code to keep code size down and reduce places where code must be changed.

B. Breaking a large program into smaller functions to avoid clutter and enable reuse and interchangeability of components.

C. Avoiding side effects by having functions return their results and change nothing else (when the problem specification allows this), to keep behavior predictable and preserve flexibility.

D. Choosing short variable names to speed development time and avoid repetitive stress injuries.

E. Include assert statements to enable preserving test cases along with the code.

When you're done, please:

- Gather up all your stuff.
- Take your stuff and your exam down to the front of the room.
- Turn in your exam; show your ID if asked.
- Exit by the doors at the front of the room. Don't go back or disturb students still taking the test.