CPSC 128

Sample Final Examination

Instructions:

- You have three (3) hours to complete the examination.
- The exam is in two parts.
- You may not use the computer while working on Part I, and must hand in Part I before turning your computer on.
- You may, but do not need to, use the computer for Part II of the exam. It is provided to enable you to use a text editor for composing programs, and to use the interpreter for testing programs (should you have the time).
- If you choose to work on a computer it is your responsibility to: 1) save your solutions to the floppy provided within the three-hour time limit; 2) backup frequently — power failures happen!

Advice:

- The value of all questions is shown — use your time wisely! (The exam will be marked out of 180 and there are 180 minutes to complete it, so the weight assigned to a question indicates the longest you should spend on it.)
- Don’t panic. Even the most daunting problem can be broken into small pieces. Stay calm and think your way through it.
- Rough work, e.g. pseudocode, can be worth substantial part marks — don’t erase or delete!
- Your code should display the elements of good programming style, e.g. meaningful variable names and precise documentation.
- You may only make use of standard library functions covered in class. If you are unsure if a particular function is allowed, ASK!
- Note carefully whether a problem requests a program, a program fragment, a function or a class and respond accordingly.
- The time allocated for the problems in Part II should be sufficient for you to sketch out correct (perhaps even elegant!) solutions. However it is probably not sufficient to produce working, debugged Python programs (of course you can always prove me wrong on this!).
- If you are unsure of anything, ASK!

Part I

What output…?

1. (10 marks) What output does the following code fragment produce?

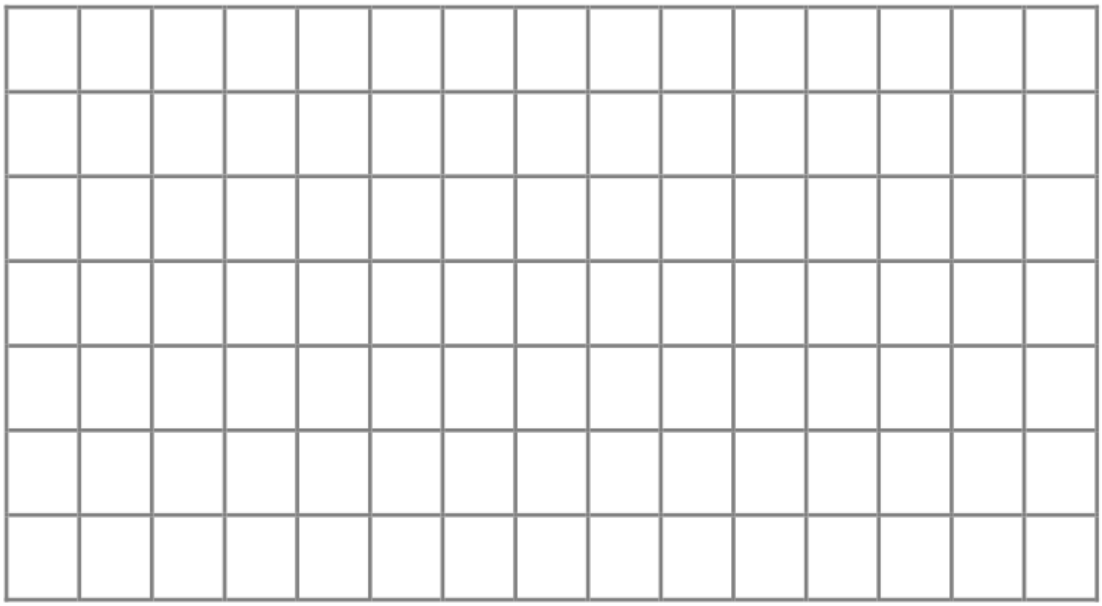
```
a = 'Tim'
b = 'Tom'
lst = [a,b]
a = 'Matt'
lst[1] = lst[0]
print(lst)
```

2. (10 marks) What output does the following code fragment produce?

```
for i in range(1,4):
    for j in range(1,4):
        print( (i//j)*(j//i), end=``)
    print()
```

3. (10 marks) What output does the following code fragment produce? (Each cell in the grid below corresponds to one character output position).

```
size = 6
for i in range(size):
    if i < size//2:
        print( i*'*')
    else:
        print( (size-i)*'*' )
```



Part II

Code Fragments
Write Python code fragments to do each of the following. (Use the names in **courier bold** as variable names or string literals).

4. (5 marks) Add the rightmost digit of **num** to **num**, e.g. if **num** is 728 add 8 to it so **num** becomes 736.

5. (10 marks) Use **if** statements to set **m** to the median (middle value) of **a**, **b**, and **c**, e.g. if **a** is 7, **b** is 1 and **c** is 4, **m** should be set to 4.

Code Fragments (continued)

For problems 6 and 7 you are given some data in the following form:

```
data = [ {'id':4721, 'sex':'F', 'age':31},
         {'id':1828, 'sex':'M', 'age':56},
         . . . lots more records . . .
         {'id':3286, 'sex':'M', 'age':29},
         {'id':5063, 'sex':'F', 'age':22}
       ]
```

6. (5 marks) Write a code fragment to display the entry of the oldest individual in the dataset, e.g. **{'id':1837, 'sex': 'F', 'age':72}**.

Code Fragments (continued)

7. (10 marks) Write a code fragment to output the number of female and the number of male records, e.g. **There are 46 females and 38 males**.

Fill in the blanks

8. (10 marks) Fill in the blanks in the program below so that it will work as the documentation specifies.

```
# Guessing game: Computer chooses a number and user tries to
# guess it.

import _________

number = random.randint(______, ______)

print("I've thought of a number between 1 and 100.")
print("Try and guess it!")
print()

guesses = __________

guess = input( "What's your guess? ")

while _____________________:

    if guess ______ number:
        print("Your guess is too high.")
    elif guess ______ number:
        print("Your guess is too low.")
    if guess ________ number:
        guess = input( "What's your next guess? " )
        __________ += 1
print("Congratulations! You guessed the number in", end=``)
print(guesses, 'guesses.')
```

Fill in the blanks

9. (10 marks) Fill in the blanks in the program below so that it will work as the documentation specifies.

```
# This function determines if a string s is a palindrome,
# i.e. if it reads the same forward and backward like 'Madam'
# or 'tot'. It ignores the case of the letters in doing so.
# The strategy is to compare the first letter to the
# last letter, then the second letter to the second last
# letter and so on, until all relevant pairs have been
# checked.

def _____(s):

    for _____ in range(_____):

        if s[index].upper() _____ s[len(s)-_____].upper():

            return _____

    return _____

isPalindrome(‘Madam’) # Should be True
isPalindrome(‘tilt’) # Should be False
```

10. (10 marks) The code in Problem 9 above contains a pair of test values to test the function. Are these sufficient? If not, what further test values would you recommend using and why?

Write a function/method/program

11. (20 marks) A shady looking character makes you the following offer. The two of you will bet on the outcome of rolling three dice. First you will look to see if there are any pairs. If there are (that is if any two of the dice have the same value) he wins. If there are no pairs and the total of the three dice is 10 or more you win. Otherwise you will roll again (and continue rolling until one of you wins). Write a program to help you determine if this is a good offer for you.

Write a function/method/program (continued)

12. (20 marks) Scrabble
Players in the game Scrabble score points by building words out of letters. The value of a word depends on the letters it is built from. (It also depends on where they are placed on the board, but we'll ignore that issue). The value of each letter is shown below. Write a function called **scrabble_score** that is passed a string and returns its Scrabble-value.

A=1	B=3	C=3	D=2	E=1	F=4	G=2	H=4	I=1	J=8
K=5	L=1	M=3	N=1	O=1	P=3	Q=10	R=1	S=1	T=1
U=1	V=4	W=4	X=8	Y=4	Z=10				

Class-based code

13. (20 marks) Add a class method called **isFullHouse** to the module **card_classes.py** given below. **isFullHouse** should return True if its Hand object instance contains five **Card**s, three with one face value and two with a second face value, and False otherwise. Be sure to indicate where your method should be inserted into the module.

```
# card_classes.py
# CPSC 128 Fall 2007

class Card:
    def __init__(self, cardnum):
        self.number = cardnum

    def face_value(self):
        return self.number % 13

    def suit(self):
        return self.number/13

class Hand:
    def __init__(self):
        self.cards = []
```

Class-based code (continued)

14. (30 marks) Web Bookmarks
a) Write the classes necessary for the code,

```
mybooks = Library( [Book("Harry Potter and the Philosopher's
stone", "J. K. Rowling"), Book("The C Programming Language",
"Brian Kernighan and Dennis Ritchie")])

yourbooks = Library( [Book("The cat in the hat", "Dr.
Seuss"), Book("Harry Potter and the Philosopher's stone",
"J. K. Rowling")] )

print(mybooks)
print()
ourbooks = mybooks + yourbooks
print(ourbooks)
```

to run and produce the output,

```
This library contains 2 books:
1. "Harry Potter and the Philosopher's stone" by J. K.
Rowling
2. "The C Programming Language" by Brian Kernighan and
Dennis Ritchie

This library contains 4 books:
1. "Harry Potter and the Philosopher's stone" by J. K.
Rowling
2. "The C Programming Language" by Brian Kernighan and
Dennis Ritchie
3. "The cat in the hat" by Dr. Seuss
4. "Harry Potter and the Philosopher's stone" by J. K.
Rowling
```