Review Problems.

Since there are no discussion sessions next week before the exam, today's session will be devoted to pre-midterm reviews.

These are typical 'exam questions' .

First, work problems 1,2,5 and 6 **by hand** first, without using your computer. Ignore problems 3,4 and 7 for now, but do try to work them at home when you are studying for the exam.

Second, check your answers: I have provided a file on the website containing the code for these problems, packaged in functions, so you can see what the code does. (For problem 1, calling this function will print out the values and the types of all the expressions.)

Finally, and most importantly: What did you get wrong? Do you see what was behind the error you made? Do you understand why the correct answer is correct, and yours was not?

1. For each of the following expressions, determine the value and the type. (For type, the answer will always be int, float, str or bool). If the question consists of a sequence of statements, give the value and the type of the expression on the last line.

```
(a) 3//2+5.0
(b) 2**3*2
(c)'ade' > 'fab'[1:]
(d) 'ba'*5+'b'
(e) x=7
    y=3
   (y*y>x) and (2*y != x)
(f)  len('abcdef')/3
(g) v = 'workbook'
    v[7]+v[1:6]+v[0]
```

2. Consider the following function definition:

```
def f(x):
     y = ''
     for c in x:
          if not (c in 'aeiou'):
               y = y+c
     return y
```

What is the result of executing the following statment?

```
print(f('University of Michigan'))
```

What would be the result if the code were written incorrectly, and `return` statement were indented one more stop (under the `if)` or two more stops (under the assignment statement `y-y+c`)?

3. (To be done at home.) The following fragment of code sets a variable `result` to a boolean value. Write a **single** boolean expression, using ands , ors and nots but no ifs, that has the same value as `result`. It might be helpful to draw a picture of the set of points where `result` is True.

```
if x>7:
     if x<=9:
          result=False
     else:
          result = True
else:
     if x<-3:
          result = False
     elif x<4:
          result =True
     else:
          result=False
```

4. (To be done at home.) Write a fragment of code (it doesn't have to be a complete function) that takes a string s, say 'oyster', and prints a pattern like the following:

```
o o o o o o
o y y y y y
o y s s s s
o y s t t t
o y s t e e
o y s t e r
```

Your code should work for any value of the initial string s.

5. The following code is meant to print out all powers of 3 less than one million (1000000), beginning with 3. Due to a small error, the actual output is not exactly as specified. What does the program really print, and how can you fix the error?

```
power = 1
 while (power < 1000000):
     power=3*power
     print (power)
```

6. Consider the function g defined below.

```
def g(n):
    count = 0
    while n%3 == 0:
           n=n//3
           count +=1
    return count
```

(a)What is the value of the variable x after execution of the statement

x=g(48)-g(36)?

(b) Give a succinct description of what exactly this function is calculating. (By this I mean, do NOT tell me what the code is doing, but rather describe the relation between the argument n and the value returned. 'The function returns the smallest prime factor of n' is a succinct description (although not correct, in this case); 'the function repeatedly divides n by larger and larger integers until a divisor is found' is what I DON'T want.

7. The following program is meant to determine the length of the shortest side of a triangle whose vertices are entered in response to the prompts. This is done by repeated calls to a function `dist` that is supposed to compute the distance between two points :

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2\,)^2}$$

There are several errors in both the function and the main program section. Find them and correct them. (Do not correct by changing to the built -in square root function; I still want you to compute the square root as the one-half power.)

```
def dist(x1,y1,x2,y2):
    xdiff = x1-x2
    ydiff = y1-y2
    diff_squares = xdiff**2 + ydiff**2
    print (diff_squares**1/2)

x1 = input('Enter the x coordinate of the first vertex.')
y1 = input('Enter the y coordinate of the first vertex.')
x2 = input('Enter the x coordinate of the second vertex.')
y2 = input('Enter the y coordinate of the second vertex.')
x3 = input('Enter the x coordinate of the third vertex.')
y3 = input('Enter the y coordinate of the third vertex.')
d1=dist(x1,y1,x2,y2)
d2=dist(x1,y1,x3,y3)
d3=dist(x2,y2,x3,y3)
print (min(d1,d2,d3))
```