1. **Short answer**

(a) The following function is supposed to test whether a given word contains a given letter. It uses a recursive strategy. Unfortunately, there are three different problems with this function. Identify these problems and rewrite the corrected function.

```python
def contains(word, letter):
    if word[0] == letter:
        return True
    elif len(word) == 0:
        return False
    else:
        contains(word[1:], letter)
```

error 1: ________________________________

error 2: ________________________________

error 3: ________________________________

(b) What output do the following lines of Python produce?

```python
x = 10
y = x * 3 + 5
x = 2 * y - 5
print("x=", x)
print("y=", y)
```

(c) Describe what makes a problem suitable for a recursive solution. What are the necessary components or characteristics of a recursive solution?
2. True or False

For each of the statements below state whether they are true or false. No justification required.

_____ If s='eat more kale' then s[1] == s[-1] would give True.

_____ Given an even-length string s, s[(len(s)//2):] would give the last half of the string.

_____ $\frac{7}{2} + \frac{7}{2} + 7 \% 2$ evaluates to 7.5

_____ Given the function below, if we were to type `print(identity(3))` and hit enter in the Python console, we would only see a single 3 displayed.

```python
def identity(num):
    print(num)
    return num
```

_____ The function `rundown(n)` produces the even numbers from n down to 2.

```python
def rundown(n):
    if n>1:
        print(n)
        rundown(n//2)
```

3. Understanding code

(a) Consider the following Python program. What output will this code produce when run?

```python
def mystery1(a, b):
    return a+b

def mystery2(a, b):
    if a > b:
        return a
    else:
        return b

def mystery3(a, b):
    if a < b:
        print(a)
        a = a + 2
        mystery3(a, b)

y = mystery1(26, 8)
print(y)

y = mystery2(26, 8)
print(y)
mystery3(16, 22)
```
(b) Consider the following function definition.

```python
def mystery(side):
    if side > 0:
        turtle.forward(side)
        turtle.left(90)
        mystery(side - 50)
```

Draw what the function above would draw on the screen if run with command `mystery(250)`. Assume the turtle initially is at the center of the window facing right. You can assume the width and height of the window are about 600 pixels. Indicate the final position and orientation of the turtle with a small triangle.

4. **Writing functions**

(a) Complete the Python program `midterm1.py`. The program asks the user for input, then calls functions `test1` and `test2` and outputs their results. You should define the (non-recursive) functions `test1` and `test2` so that the provided main program (at the bottom of the box below) will work without any modification. Do not use any pre-defined mathematical functions, rather simply use multiplication, addition, conditionals, etc, to complete the code. Here is an example of how your program should work (the user’s input is the -7 in bold). Note: The function calls and input/output are done for you; you do not need to rewrite the main program, just write the function definitions for `test1()` and `test2()`.

```python
>>> runfile('midterm1.py')
Enter a number: -7
square of -7 is 49
absolute value of -7 is 7
```

```python
# midterm1.py

# define functions test1 and test2 here:

# main program that calls test1 and test2:

x = int(input("Enter a number: "))
print("square of", x, "is", test1(x))
print("absolute value of", x, "is", test2(x))
```
(b) Write a recursive function called `every_other_letter` that takes a string parameter as input and returns a string that contains every other letter in the string starting with the first letter. For example:

```python
>>> every_other_letter('banana')
'bnn'
>>> every_other_letter('computer')
'cmue'
>>> every_other_letter('kale')
'kl'
>>> every_other_letter('a')
'a'
```

[Note: a concise correct answer can be given in just 5 lines of code. Just give the function definition, not the main program, and do not use `print()` or `input()`.

5. **Binary numbers**

For each of the following (a)-(c), perform the conversion. Use 16-bit unsigned binary numbers.

a. 372 from decimal to (unsigned) binary

b. 1024 from decimal to (unsigned) binary

c. 0000 0000 0000 1111 from (unsigned) binary to decimal

For each of the following (d)-(f), perform the conversion. Use 16-bit two’s complement numbers.

d. 0000 0000 0001 0101 from (two’s complement) binary to decimal

e. 1111 1111 0111 1101 from (two’s complement) binary to decimal

f. -1024 from decimal to (two’s complement) binary

g. B8 from hexadecimal to (two’s complement) binary, then to decimal