Question 1. [10 MARKS]

The following code runs without errors:

```
def func1(n):
    n = n * 2
    return n

def func2(s):
    s = s + ", really"

def func3(l):
    l[0]=1

def func4(l):
    l = [1,2,3]
    return l

def func5(l):
    l = l.append(4)
    return l

if __name__ == "__main__":

    number = 10
    number = func1(number)
    print number

    s = 'I like trees'
    func2(s)
    print s

    list1 = [0,0,0]
    func3(list1)
    print list1[0]

    list2 = [4,5,6]
    func4(list2)
    print list2

    list3=[7,8,9]
    func5(list3)
    print list3
```

On the following page, show the five lines of output that this code produces. **Strong hint:** Use the blank space provided to trace the code using the memory model.

Solution:

```
20
I like trees
1
[4, 5, 6]
[7, 8, 9, 4]
```

Question 2. [5 MARKS]

Each of these subquestions contains a block of code. Treat each block of code independently (code in one question is not related to code in another), and fill in the blanks for each question.

Part (a) [1 MARK] Order of Execution

```
var_A = [5]
var_B = var_A
var_A[0] = 10
```

After this code is executed, the value of `var_B` is _____.

Solution:

`[10]`

Part (b) [2 MARKS] **Conditionals and Booleans**

The table to the right shows one way in which one might decide when to buy groceries. If you have an empty fridge, then you need to buy groceries, as is the case if you haven't bought groceries for a week in which case you may need to replenish some of the things that go bad. Assume that you have an `int` variable `last_groceries` that tells you how many days it's been since you've bought groceries and a `bool` variable `empty` that is True iff the fridge is empty. Fill in the boolean conditions in the code below to determine what to print to the screen.

	Time from last grocery run	
Fridge empty	< 7 days	>= 7 days
Empty	Buy Groceries	Buy Groceries
Not Empty	Wait	Buy Groceries

```
if ____________________________________________________________:
    print 'Buy Groceries'

elif ____________________________________________________________:
    print 'Wait'

else:
    print 'Buy Groceries'
```

Solution:

```
if empty:
    print 'Buy Groceries'

elif last_groceries < 7:
    print 'Wait'

else:
    print 'Buy Groceries'
```

Part (c) [1 MARK] Data Types

Fill in the blank so that when this code is run, the user is asked to enter their weigh in kg and height in m. Their BMI is then printed. The BMI is: (their weight in kg)/(their height in m)^2. The user input may contain decimal values (e.g., `1.75`).

```
num1 = raw_input("Please enter your weight in kg: ")
num2 = raw_input("Please enter your height in m: ")

print "Your BMI is " , ________________________________________
```

Solution:

```
float (num1)/float (num2)**2
```

Part (d) [1 MARK] Calling Functions

Fill in the blank to call `average_rainfall` to obtain the average rainfall of Moose Jaw in 2010.

```
def average_rainfall(city, year):
    '''Return the average rainfall of str city in int year.'''
    # The code for this function is not shown.
    return average

print "In 2010, the average rainfall in Moose Jaw was" , ___________________________________
```

Solution:

```
average_rainfall ('Moose Jaw', 2010)
```

Question 3. [7 MARKS]

Write the following function according to its docstring.

```
def evens_and_odds (L):
  '''L is a list. Return a tuple whose first element is
  a list of the elements of L with even indices, and whose second
  element is a list of the elements of L with odd indices. Forexample
  evens_and_odds([1,2,3,4,5,4,3]) should return
  ([1,3,5,3], [2,4,4])'''
```

Solution:

```
def evens_and_odds(l):
    '''l is a list. Return a tuple whose first element is
    a list of the elements of l with even indices, and whose second
    element is a list of the elements of l with odd indices. For example
    evens_and_odds([1,2,3,4,5,4,3]) should return
    ([1,3,5,3], [2,4,4])
    '''
    evens, odds = [],[]
    for i in range(len(l)):
        if i % 2 == 0:
            evens.append(l[i])
        else:
            odds.append(l[i])
    return (evens, odds)
```

Question 4. [8 MARKS]

Write the function below, according to its docstring. You must not use a for-loop in this question or your solution will earn zero.

```
def first_float(L):
    '''L is a list of floats.  Return the index of the first element of L that
    has a non-zero component after the decimal. (ie. the first element
    that does not represent an integer). If there is no such element, return -1.
    For example first_float([1.0, 2.0, 1.0, 1.5, 2.0]) will return 3.'''
```

Solution:

```
    i = 0
    while i < len(L) and L[i] == int(L[i]):
        i += 1
    if i < len(L):
        return i
    else:
        return -1
```

Question 5. [7 MARKS]

Complete the following function according to its docstring description.

```
def ints_only(d):
  '''d is a dict whose values can be of any type.
  Return a new dictionary which contains only the key
  value pairs from d whose values are of type int.
  For example, ints_only({5:[2,3], 4:5, 20: '1'}) would
  return {4:5}'''
```

Solution:

```
def ints_only(d):
  '''d is a dict whose values can be of any type.
  Return a new dictionary which contains only the key
  value pairs from d whose values are of type int.
  For example, ints_only({5:[2,3], 4:5, 20: '1'}) would
  return {4:5}'''

  d_ret = {}

  for k in d:
    if type(d[k]) == int:
      d_ret[k]=d[k]
  return d_ret
```

Question 6. [8 MARKS]

Write the function below, according to its docstring.

```
def goal_scorers(filename):
    '''str filename is the name of a file that stores the number of goals scored by each
    player in a season. Each goal total is stored in a single line as an amount followed by
    the character g (for example: 34g). Return the number of players that scored more than
    20 goals in a season.'''
```

Solution:

```
    goals_scored = open(filename, 'r')
    total = 0
    for line in goals_scored:
        if int(line[:-1]) > 20:
            total += 1
    return total
```