Question 1. [6 MARKS]

Consider the function `square_sequence` below:

```
def square_sequence(s):
    '''Given a sequence s of numbers, return the same sequence except that
    each value is replaced by its square.'''

    for i in range(len(s)):
        s[i] = s[i] * s[i]
    return s
```

What will be printed after each of the following code fragments is executed? If the code fragment runs correctly, state what will be printed in the space provided. If you think the fragment produces an error, say so, and explain why it fails.

```
s1 = range(0, 15, 3)
print square_sequence(s1)
```

```
[0, 9, 36, 81, 144]
```

```
s2 = (-1, 3, -5, 7)
print square_sequence(s2)
```

There will be an error, because `square_sequence` is attempting to modify a tuple. Tuples cannot be modified.

```
s3 = {3: 1, 7: 0, 'a': 2}
print square_sequence(s3)
```

```
There will be a Key Error because 0 is not a key in the dictionary.
```

Question 2. [6 MARKS]

Complete the following function according to its docstring description. Assume that all coordinates `(x, y)` are within the bounds of the picture.

```
import media

def add_lines(pic, lines_dict):
    '''Add the lines specified in dictionary lines_dict to the picture pic.
    lines_dict contains coordinate tuples as keys.  The values in lines_dict
    are tuples containing a list of coordinate tuples and a colour. The lines
    specified in a key:value pair in the dictionary go from the point specified
    in the key to each of the points given in the coordinate list in the value.

    For example, given the following key:value pair from the dictionary:
        (3, 5): ([(4, 6), (2, 9)], Color(255, 255, 255))
    Add a line in pic from point (3, 5) to (4, 6) and a line from (3, 5)
    to (2, 9).  Both lines should be coloured Color(255, 255, 255).'''

    for coord1 in lines_dict:
        color = lines_dict[coord1][1]
        for coord2 in lines_dict[coord1][0]:
            media.add_line(pic, coord1[0], coord1[1], coord2[0], \
                           coord2[1], color)
```

Question 3. [8 MARKS]

Write a program that prompts the user for a Python file using `choose_file`, opens the file, creates a dictionary containing the names and parameters of all functions in the file using function `find_functions`, and then prints the dictionary. This program consists of two parts: a function, `find_functions`, and a main block at the bottom of the next page.

For this Python file:

```
def f():
    pass

def g(x, y):
    pass
```

Your program should print:

```
{'f': [], 'g': ['x', 'y']}
```

The header for `find_functions`, including a docstring describing its behaviour, is included below. Assume that all function definitions start with the string "def". Also assume that you have been given a function named "parse_func_def" (from some module called "function_parser") that can break apart a line containing a function definition. Use "parse_func_def" in `find_functions`. Here is the help information from `help(function_parser.parse_func_def)`:

```
parse_func_def(line)
    Return a tuple (func_name, func_parameters) where func_name is the name of the function
    defined in the string line and func_parameters is a list of strings representing the
    parameters of the function defined in line.
```

```
-------------------------------------------------------------------------------
import media
import function_parser

def find_functions(f):
    '''Return the dictionary containing all of the functions from the open Python file f.
    For each function, the dictionary should contain the function's name as a
    key and a list containing the names of its parameters as a value.'''



    func_dict = {}
    for line in f:
        if line.startswith("def "):
            func_name, params_list = function_parser.parse_func_def(line)
            func_dict[func_name] = params_list

    return func_dict
```

```
if __name__ == "__main__":


    python_file = open(media.choose_file())
    func_dict = find_functions(python_file)
    print func_dict
```