Question 1. [2 MARKS]

Part (a) [1 MARK] What is the output of the following?

```
pic = media.create_picture(50, 100)
pic2 = media.add_text(pic, 0, 0, 'test', media.yellow)
print type(pic2)
```

`None or NoneType`

Part (b) [1 MARK] Rewrite the following code without an `if`-statement.

```
if ketchup and not mustard:
    return True
else:
    return False
```

`return ketchup and not mustard`

Question 2. [2 MARKS]

In each question below, fill in the box with python code that will make the program behaviour match the comments. You may **not** make any other changes to the code.

Part (a) [1 MARK]

```
name = 'Matthew'
age = 3

# Print the following: Matthew is 3!

print '%s is %d!' % (name, age)
```

Part (b) [1 MARK]

```
pic = media.load_picture(media.choose_file())

# get the pixel at (10, 4)
```

`media.get_pixel(pic, 10, 4)`

```
# set the pixel at (10, 4) to yellow
media.set_color(pix, media.yellow)
```

Question 3. [8 MARKS]

Part (a) [4 MARKS] Complete the following function according to its docstring description.

```
def change_green(pic, factor):
    '''(Picture, float) -> Picture
    Return a new picture that is a copy of pic, but with each pixel's green color
    component set to its original value multiplied by factor. factor is a value
    between 0.0 and 1.0, inclusive.'''

    new_pic = media.copy(pic)

    for pixel in new_pic:
        green = media.get_green(pixel)
        new_green = int(green * factor)
        media.set_green(pixel, new_green)

    return new_pic
```

Part (b) [4 MARKS]

Write a main block that allows the user to choose a file, prompts the user with, '`Enter a value between 0.0 and 1.0, inclusive: `', applies the `change_green` function from part (a) to the picture in that file using the value entered by the user, and displays the resulting picture. You may assume that the user chooses a valid picture file and enters a valid value.

```
if __name__ == '__main__':

    pic = media.load_picture(media.choose_file())
    factor = float(raw_input('Enter a value (between 0.0 and 1.0): '))
    new_pic = change_green(pic, factor)
    media.show(new_pic)
```

Question 4. [8 MARKS]

Consider the following two .py files, which are saved in the same directory (folder).

module_a.py:

```
def f(s):
    result = ''

    for char in s:
        if char == char.upper():
            result = result + char

    return result

if __name__ == '__main__':
    print f('EFg')

# this code is not inside the
# body of the if-statement
print f('aBcde')
```

module_b.py:

```
import module_a

def g(s):
    answer = module_a.f(s)
    return len(answer)

if __name__ == '__main__':
    print module_a.f('WXyZ')
    print g('TeSTiNg')
```

This question continues on the next page. You may use the space below for rough work.

Part (a) [1 MARK]

How many lines of output are produced when `module_b` is executed (by clicking Run)?

Circle one: 2 lines **3 lines** 4 lines

Part (b) [4 MARKS]

In the table below, show the output from running `module_b`. If there are fewer than four lines of output, leave the unused box(es) empty.

B
WXZ
4

Part (c) [3 MARKS]

Write a good docstring for the function `f` from `module_a`.

```
(str) -> str
Return a new string that contains the uppercase letters from s.
```