University of Cape Town

Department of Computer Science

Computer Science CSC1010H

Class Test 2

Wednesday, 20 August 2014

Marks: 35

Time: 40 minutes

- Approximate marks per question are shown in brackets
- The use of calculators is permitted

Surname Initials

NAME: Mtsweni Edward E.S.

STUDENT NO: MTSEDW004 **COURSE CODE:** CSC1010H

This paper consists of 6 questions and 6 pages (including this cover page).

Mark Allocation							
Question	**Marks**	**Internal**	**External**	**Question**	**Marks**	**Internal**	**External**
1	5			5	4		
2	7			6	6		
3	6						
4	7						
	Total				**Total**		
					Grand Total		
					Final Mark		

Internal Examiner:	**External Examiner:**

Question 1. [5 marks]

Consider the following problem. Answer it appropriately.

The Petersens have recently moved to a new town and are arranging a surprise birthday party for their son Andre, and have invited three families from the neighbourhood, the Smiths, the Januarys and the Hectors. They plan to make up party packets for the kids to take home after the party, blue for boys and pink for girls.

Being super organised, Mrs Petersen with the help of Mr Petersen wants to determine how many of each colour party packet she needs to buy, and also how many of each colour she needs to put aside for each family.

They sit down and come up with the following information. Mrs Petersen remembers that the Hectors have a "pigeon pair", i.e. a boy and a girl. Mr Petersen recalls that the Januarys only have a set of identical twin boys. Mrs Petersen notes that she's only ever noticed two girls from these local families to come over to play. Mr Petersen notes that the Smiths have three children, since the family fits nicely into their family sedan when they go out.

You happen to be visiting the Petersens at this point, and want to impress them with the problem solving skills you've learnt at university. Using the information they've provided, determine how many of each colour party packet they need to buy and how many of each colour they need to allocate to each family and what the total number of party packets are.

Use a diagram to show how you solve the problem.

	Boys	Girls
	Blue	Pink
Smiths	2	1
Januarys	2	0
Hectors	1	1

The Hectors have a boy and a girl, meaning 1 blue & 1 pink party packets. The Januarys have 2 twin boys, meaning 2 blue packets. Since there are 2 girls from these families, this means that the Smiths have a daugher & 2 boys. 5 blue and 2 pink packets must be bought, with 2 blue & 1 pink going to the Smiths, 2 blue going to the Januarys & 1blue2 and 1 pink going to the Hectors [5]

Question 2. [7 marks]

Answer the following questions:

a) When using debugging features in an IDE, what should the user typically do once execution has reached the breakpoint?

The user should step over the code, do user input/output and watch how variables change in the stack window [2]

b) When a new module has been defined, how do you ensure that it is accessible and can be imported into a program with no problems, i.e. "import newmodule" works?

By saving the file in the Python/Lib Directory [1]

c) Explain what happens in memory when Python makes successive recursive function calls.

run-time stack overflow occurs, meaning insufficient memory [1]

Indicate whether the following statements are True or False.

d) The accepted Python coding convention for module names is long descriptive names in uppercase.

False [1]

e) Curly brackets {} are used to enclose parameters to a function.

false [1]

f) The print() function can be used to write to a file.

True [1]

Question 3. **[6 marks]**

Write a Python function called draw_line() which draws a horizontal line of characters. The draw_line() function should take two parameters, with the first being the size of the line (i.e. the number of characters) and the second parameter being the character with which to draw the line. This character parameter should have a default value of an asterisk ('*').

Calling the draw_line() function with the following parameters should produce the corresponding output:

```
draw_line(5)          produces        *****
draw_line(6,'$')                      $$$$$$
```

```
def draw_line(size, character):
    intg_size = int(size)
    if character == "":
        print("{}".format("*" * intg_size))
    else:
        print("{}".format(character * intg_size))

draw_line(4, "#")
```

[6]

Question 4. **[7 marks]**

Consider the following recursive function definition:

```
def do_this(stuff):
    if len(stuff) == 0:
        return ""
    else:
        return str(stuff[0] * 2) + do_this(stuff[1:])
```

a) What datatype can the parameter to this function be?

The datatype of the parameter can be a string or tuple [2]

b) What is the base case for this function?

If the length of the parameter is zero, it'll return an empty space, which is the base case [1]

c) Based on the `do_this()` function definition, what will the following statements display?

i. `print(do_this([1,2,3]))`

223 [2]

ii. `print(do_this("123"))`

~~223~~ 1123 [2]

Question 5. [4 marks]

Consider the following Python program and answer the questions below:

```
def main():
    f = open('to_do_list.txt','a')
    while True:
        thing_to_do = input('Enter thing to do:')
        if thing_to_do == 'done': break
        else:
            f.write(thing_to_do + '\n')
    f.close()

main()
```

a) What is the name of the file created?

to_do_list [1]

b) What mode is the file created in?

append mode [1]

c) Looking at the code, how does the user terminate the program?

By typing in 'done' [1]

d) How will the information that the user enters be written in the file?

The information will be written on a new line everytime the user enters input [1]

Question 6. [6 marks]

Consider the following definition of the *classify_weight()* function. Specify test cases which thoroughly test the function, using equivalence classes and boundary value. For each test case specify whether it is an equivalence class value or a boundary value.

```
# classifies weight in kgs
def classify_weight(w):
    if 0 < w <= 60:
        return "light"
    elif 60 < w <= 120:
        return "heavy"
    else:
        return "error"
```

- Equivalence classes
  * category values: 20, 109
  * erroneous values: -1, 0

- Boundary values
  * on boundary: 60, 120
  * below boundary: -1, -2
  * above boundary: 122, 170

[6]