

Daniel Newman - Marks

Q1 a) Algorithm for tea: 15 /17

b) Different drink selections - 5/5

c) Storing data on a computer - 3 /3 **Total for Question 1 - 23/25**

Q2 Flowchart 15/25 **Total for Question 2 - 15 /25**

Q3 Python code for 2) above **Total for Question 3 - 25 /25**

Q4 a) - An oval in turtle - 0/9

b) A 5 point star in turtle - 8/8

c) A spiral in turtle - 6/8 **Total for Question 4 - 14/25**

OVERALL TOTAL - 77%

Mark agreed

77%

Computational Thinking Test Oct 2017

Name Daniel Newman

Student number 1702330

Programme Python Web Development

Answer only the questions you can – if you cannot answer a question, move on to the next question. This test is not negatively marked, so you may wish to guess if unsure. Questions 1 and 2 to be answered on this paper, questions 3 and 4 to be answered using Python and saved (then uploaded via Moodle for staff).

1) You have been selected to program a new robot intended to create hot drinks. The robot is capable of following simple, tea and coffee-oriented commands precisely, but has no understanding either of the process, or the fundamental principles which underpin it (e.g. that a kettle requires power). The robot has access to the following items:

- Kettle (initially unplugged)
- Tea bags
- Jar of ground instant coffee
- 1L carton of milk
- Unopened bag of sugar
- 1 metal tea spoon
- 1 large mug
- Access to a sink for water and an electrical socket for power

a) You are required to give the robot the set of instructions (algorithm) necessary to successfully make a cup of milky coffee with 1 teaspoon of sugar. Each instruction should be on a new line. Ensure that your instructions are in a logical order and no steps are missed; while highly capable of following instructions, the robot cannot solve problems independently.

(17)

1. Check all items are available.
2. Fill large mug with tap water. (fill it half full)
3. Pour water-filled mug into kettle. 3
4. plug kettle into socket and activate it.
5. place the now empty mug under the kettle
6. wait for mug to fill ✓1
7. X pour 5 teaspoons worth of instant coffee into mug. ✓2
8. wash teaspoon and the open bag of sugar
9. fill teaspoon with sugar and pour into mug. ✓3
10. fill the other half of mug using the 1L milk (do not overflow) ✓2
11. Stir with tea spoon ✓1
12. Place cup of coffee on table.

+2

(15)

(PTO for more space)

- b) The robot is, of course, capable of making many variations of hot drink. The user must be permitted to give information about their drink preferences to the robot before it is created.

1. What drink would the user want?

Coffee or tea ✓

2. How much sugar / Sweeteners

1 or 2 or 3 or 4 or 5 teaspoons ✓

3. How much milk?

low or medium or high. (5) ✓

(5)

- c) If the instructions were converted to code, explain briefly how this data may be stored.
What is the name given to a piece of data stored by a computer?

The data would be stored briefly until the robot is finished. It would be stored as a variable which the robot would act on differently depending on the request.

(3)

2)

We have been given the requirements for a program which are:

Requirement 1 - Says "Hello" to the user at the start.

Requirement 2- Reads a series of numbers from the user, adding them together.

Requirement 3 - Stops when the user inputs -1 as a value.

Requirement 4 - Once the user elects to stop inputting numbers, outputs the sum of the numbers input.

Requirement 5 - Says "Goodbye" to the user.

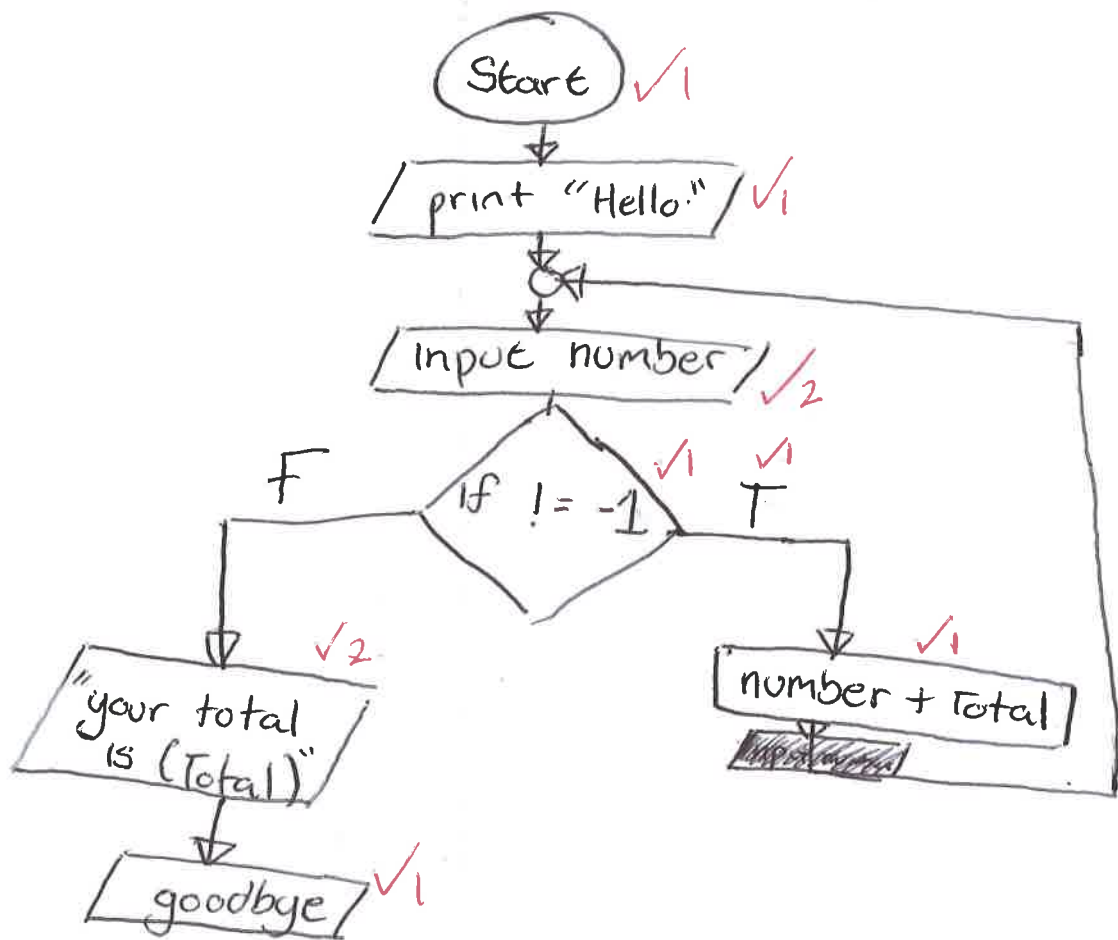
For example, the program may run something like:

```
Hello
Enter number:
8
Enter number:
10
Enter number:
3
Enter number:
-1
Your total is 21.
Goodbye.
```

Of course, the user may enter any set of numbers and get a valid result, here's another example of the program running:

```
Hello
Enter number:
9
Enter number:
-1
Your total is 9.
Goodbye.
```

Write the flowchart for this program:



+ 2 box
+ 2 Branch
+ 1 logic

15

- 3) Now write the in Python code for the program given in 2) above. Remember, the output might look similar to:

```
Hello.  
Enter number (-1 to exit): 8  
Enter number (-1 to exit): 10  
Enter number (-1 to exit): 3  
Enter number (-1 to exit): -1  
Your total is 21  
Goodbye.
```

(15)

- (Advanced) The program should additionally prevent the user from adding negative numbers (other than -1 to exit). For example:

```
Hello.  
Enter number (-1 to exit): 5  
Enter number (-1 to exit): 10  
Enter number (-1 to exit): -5  
You cannot input a negative number.  
Enter number (-1 to exit): 7  
Enter number (-1 to exit): 2  
Enter number (-1 to exit): 6  
Enter number (-1 to exit): -1  
Your total is 30  
Goodbye.
```

(10)

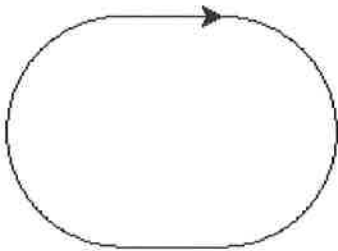
4) Using Python with turtle, draw the following shapes:

a) An oval. Remember that the code to draw a circle is:

```
import turtle

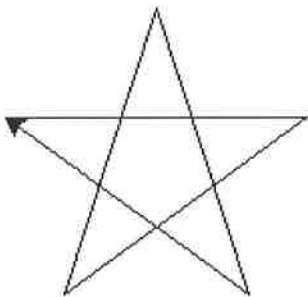
count = 0
while (count < 360):
    turtle.forward(1)
    turtle.right(1)
    count = count + 1
```

You should start with this as a base, and modify it to form an oval similar to:



(7)

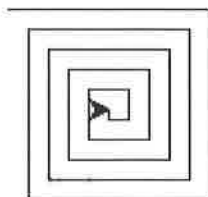
b) A 5 point star. This can be achieved without lifting the pen. It should look similar to:



Hint: the angle of rotation at the end of each point is 144 degrees.

(8)

c) A spiral. This may be a square spiral similar to:



You should not equate each edge to a line of code. Instead consider what you can use to reduce the amount of code you are required to produce.

(10)

Test October 2017 Question 3a and 3b (1).py

#Daniel Newman

```
print("Hello")
Answer = 0
```

```
Total = 0
```

```
while Answer != -1:
```

```
    if Answer < -1:
```

```
        print("You cannot enter negative numbers other than -1")
```

```
        Answer = int(input("Enter Number. (enter -1 to exit) "))
```

```
    if Answer >= 0:
```

```
        Total = Total + Answer
```

```
        Answer = int(input("Enter Number. (enter -1 to exit) "))
```

```
else:
```

```
    print("Your total is {}".format(Total))
```

```
    print("Goodbye")
```

✓ Good.

✓

Turtle Test October 2017 4a (1).py

#Daniel Newman

import turtle

count = 0

while(count < 360):

turtle.forward(9)

turtle.right(4)

turtle.forward(6)

count=count+1

Turtle Test October 2017 4b (1).py

#Daniel Newman

import turtle

```
turtle.forward(300)
turtle.right(144)
turtle.forward(300)
turtle.right(144)
turtle.forward(300)
turtle.right(144)
turtle.forward(300)
turtle.right(144)
turtle.forward(300)
turtle.right(144)
turtle.forward(300)
turtle.right(144)
```



Turtle Test October 2017 4c (2).py

#Daniel Newman

```
import turtle  
count = 300
```

```
while count <=300:  
    turtle.forward(count)  
    turtle.right(90)  
    count = count - 5
```

