GCSE Computer Science (9-1)

EDULiTO

Producing Robust Programmes

Topic Tests

Photocopiable Resources

Terms and Conditions of Use

Topic Test - Producing Robust Programs

1 (a) In terms of computer programming, what is meant by defensive design and why is it used? [2]

………

………

………

………

1(b) The table below is about types of defensive design. Complete the table. [10]

Term	Definition	Example
Validation		
Sanitization		
Planning for contingencies		
Anticipating misuse		
Authentication		

2 A common defence strategy is to ensure that code is written in such a way that other programmers can understand it. This is called maintainability and it reduces the chance of coding mistakes or bugs. Describe two ways in which code can be made more maintainable. [2]

1………

2………

3 (a) In computer programming, why is it important that programs are tested? [1]

………

………

3 (b) Complete the table below. [2]

Types of testing	Explanation
	Testing is ongoing throughout the development process. You may code an aspect of your program and test it before moving on.
	This is carried out at the end of the development process, when the program is complete, the program should be tested again (as a whole) against the requirements of the user to ensure their needs have been met.

4. (a) Fill in the gaps below using the words shown. [6]

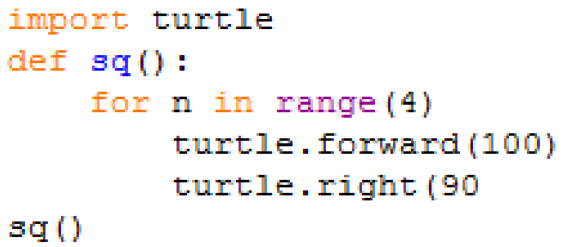
logic programming unexpected syntax translated convert

A ………………………… error is simply an error where the code written doesn't meet the rules of the ………………………… language.

These errors appear when the source code is …………………………into machine code. The translator tries to …………………………the code, but if the code doesn't meet the rules of the translator, it throws up an error.

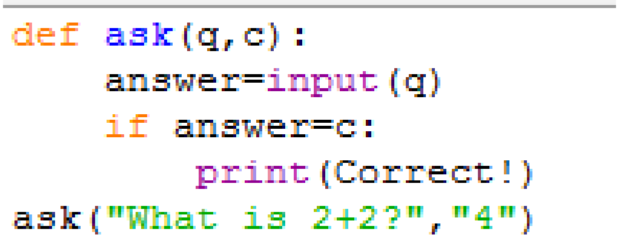
A …………………………error is one where the code is written in accordance with the programming rules and is therefore translated and runs, however, the program produces …………………………results.

4(b) Circle the syntax errors in the Python program shown below. [2]



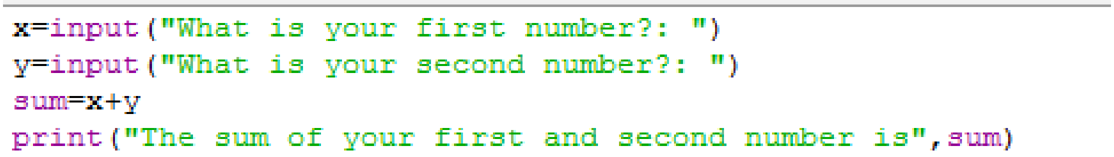
```
import turtle
def sq():
    for n in range(4)
        turtle.forward(100)
        turtle.right(90
sq()
```

4(c) Circle the syntax errors in the Python program shown below. [3]

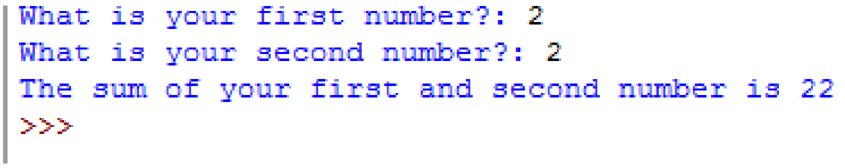


```
def ask(q,c):
    answer=input(q)
    if answer=c:
        print(Correct!)
ask("What is 2+2?","4")
```

5(a) What is the logic error in this Python program used to calculate the sum of two numbers? [1]



```
x=input("What is your first number?: ")
y=input("What is your second number?: ")
sum=x+y
print("The sum of your first and second number is",sum)
```

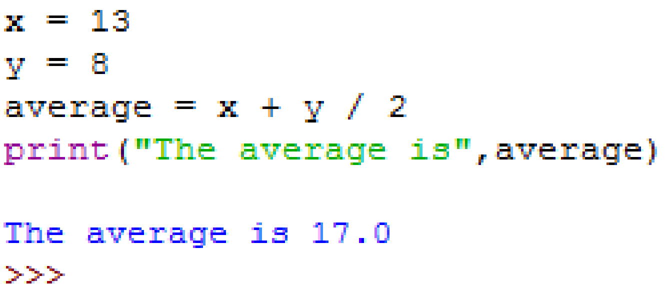


```
What is your first number?: 2
What is your second number?: 2
The sum of your first and second number is 22
>>>
```

………

………

5 (b) What is the logic error in this Python program used to find out the average of two numbers? [1]



```
x = 13
y = 8
average = x + y / 2
print("The average is",average)

The average is 17.0
>>>
```

……

……

6 (a) Describe the following types of test data and given an example for each. [6]

Type of test data	Description	Example
Valid Data		
Invalid Data		
Borderline Data		

6(b) List four things that must be included in a test plan. [4]

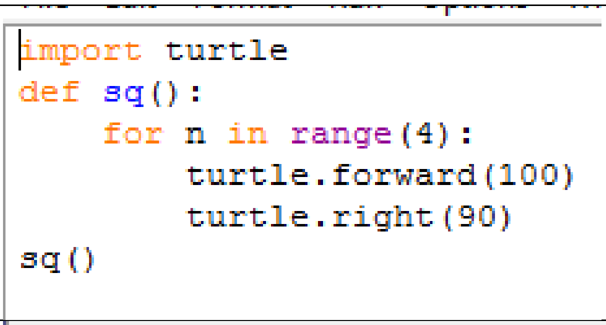
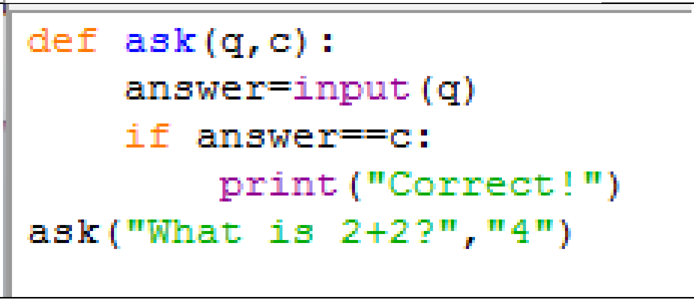
1……

2……

3……

4……

<table>
<tr><th colspan="4">Topic Test Producing Robust Programs - Mark Scheme</th></tr>
<tr><th>Question Number</th><th>Answer</th><th>Additional Guidance</th><th>Mark</th></tr>
<tr><td><b>1 a</b></td><td>Defensive design is used to ensure that a piece of software functions under any circumstances.[1]
It is used where it is important that software is available all of the time and is secure. [1]</td><td></td><td><b>2</b></td></tr>
<tr><td><b>1 b</b></td><td><table>
<tr><th>Term</th><th>Definition</th><th>Example</th></tr>
<tr><td>Validation</td><td>Checks if the input meets a set of criteria</td><td>Anything suitable</td></tr>
<tr><td>Sanitization</td><td>Modifies the input to ensure that it is valid</td><td>Anything suitable</td></tr>
<tr><td>Planning for contingencies</td><td>Preparing for a possible future issue with hardware or software.</td><td>Anything suitable</td></tr>
<tr><td>Anticipating misuse</td><td>Ensuring that the computer system is prepared for the misuse of the system by external sources.</td><td>Anything suitable</td></tr>
<tr><td>Authentication</td><td>Ensuring that the system is only accessible to users who pass a security test.</td><td>Anything Sensible</td></tr>
</table></td><td>1 mark for each correct definition and1 mark for each suitable example.</td><td><b>10</b></td></tr>
<tr><td><b>2 a</b></td><td>Comments [1]
Indentation [1]
Formatting [1]
Max 2</td><td></td><td><b>2</b></td></tr>
<tr><td><b>3 a</b></td><td>Testing is required to make sure that that a program functions correctly. [1]
Meets the needs of the end user. [1]
Max 1</td><td></td><td><b>1</b></td></tr>
<tr><td><b>3 b</b></td><td><table>
<tr><th>Types of testing</th><th>Explanation</th></tr>
<tr><td>Iterative Testing [1]</td><td>Testing is ongoing throughout the development process.
You may code an aspect of your program and test it before moving on.</td></tr>
<tr><td>Final/Terminal Testing
[1]</td><td>This is carried out at the end of the development process, when the program is complete, the program should be tested again (as a whole) against the requirements of the user to ensure their needs have been met.</td></tr>
</table></td><td></td><td><b>2</b></td></tr>
<tr><td><b>4 a</b></td><td>A <b>syntax</b> error is simply an error where the code written doesn’t meet the rules of the <b>programming</b> language.
These errors appear when the source code is <b>translated</b> into machine code. The translator tries to <b>convert</b> the code, but if the code doesn’t meet the rules of the translator, it throws up an</td><td></td><td><b>6</b></td></tr>
</table>

<table>
<tr><td></td><td>error.
A <b>logic</b> error is one where the code is written in accordance with the programming rules and is therefore translated and runs, however, the program produces <b>unexpected</b> results.</td><td></td><td></td></tr>
<tr><td><b>4b</b></td><td>
import turtle
def sq():
    for n in range(4):
        turtle.forward(100)
        turtle.right(90)
sq()</td><td></td><td><b>2</b></td></tr>
<tr><td><b>4 c</b></td><td>
def ask(q,c):
    answer=input(q)
    if answer==c:
        print("Correct!")
ask("What is 2+2?","4")</td><td></td><td><b>3</b></td></tr>
<tr><td><b>5 a</b></td><td>Datatype is <b>str</b> and should be <b>int</b></td><td></td><td><b>1</b></td></tr>
<tr><td><b>5 b</b></td><td>Should be (x+y)/2</td><td></td><td><b>1</b></td></tr>
<tr><td><b>6 a</b></td><td><table>
<tr><th>Type of test data</th><th>Description</th><th>Example</th></tr>
<tr><td>Valid Data</td><td>The data should produce the expected result.</td><td>Anything suitable</td></tr>
<tr><td>Invalid Data</td><td>The data should produce an error.</td><td>Anything suitable</td></tr>
<tr><td>Borderline Data</td><td>It is important to test that data on the edge between valid and invalid are dealt with correctly by the program.</td><td>Anything suitable</td></tr>
</table></td><td></td><td><b>6</b></td></tr>
<tr><td><b>6 b</b></td><td>Test number [1]
Test data [1]
The reason for the test [1]
The expected outcome of the test [1]
The actual result of the test [1]
Changes required to the program [1]
Max 4</td><td></td><td><b>4</b></td></tr>
</table>