NAME:

Errors

There are three types of errors you can get in your code:

- Syntax
- Logic (semantic)
- Run Time

Explain each of these errors and state how they would manifest (make themselves known) to the programmer/user.

Syntax- Usually something like a spelling mistake or a missing colon which appears to the user, saying syntax error.

Logical- An output or solution is output that isn't expected. Usually something is incorrect but not the code. The code will work, it's just something like maths done wrong by the programmer. For example if they accidently wrote 5+5 instead of 5-5, the code would just run it anyway.

Run time- This is an error caused when you may be trying to link to files, images or other things in libraries. For example, you can import things like math or random but if you tell the program to look in a specific library and it isn't there, your code could break or crash.

Errors in your Pseudocode/Flowchart Algorithm

Only one of the errors above can exist in your pseudocode/flowchart, which one is it and why?

Logical errors because the algorithm might not produce the expected output or solution. Syntax doesn't really matter because pseudocode is just structured English but logical errors mean the programmer has entered something wrong probably by accident.

Planning

There are methods, things you can do to check your code before you even start to program. These are writing a **Dry Run** on paper by drawing a **Trace Table**. Basically, all you do is create a table with the variable names across the top and write in the values you would expect the variables to take through each line of the code.

Below is some pseudocode to create a list of the 2 times table and the trace table you would plan out to see if you had completed the logic correctly.

```
FOR index = 2 to 2
        print (“This is the “ + index “times table.”)
        FOR times = 2 to 12
                Print (“times + “X” + index + “ = “ + index * times)
        NEXT times
NEXT index
```

index	Output	times	Output
2	2	1	2
2	2	2	4
2	2	3	6
2	2	4	8
2	2	5	10
2	2	6	12
2	2	7	14
2	2	8	16
2	2	9	18
2	2	10	20
2	2	11	22
2	2	12	24

Task 1

Look at the pseudocode below and complete the Trace Table.

```
turns
x = 3
WHILES turns < 22
        x = x * 3
        turns = turns + 3
END WHILE
print (x)
print (turns)
```

turns	x	Output
0	3	3, 0
3	9	9, 3
6	27	27,6

Task 2

Look at the pseudocode below and complete the Trace Table.

```
number_1 = 3
number_2 = 2
total = 0
FOR x = 1 to 5
        number_1 = number_1 * x
        number_2 = number_2 * (x + 1)
        total = number_1 + number_2
NEXT x
print (total)
```

number_1	number_2	total	Output
3	2	0	0
3	4	7	7
6	12	18	18
18	48	66	66

72	240	312	312
360	1440	1800	1600

Task 3

Complete work from other programming or theory booklets that are overdue or deadline approaching.

Task 4

Program in Python the code for all three examples above:

- Two times table

```
index = 2
print("This is the", index, "times tables.")
times = 2
while times <=12:
    print(index, "x", times, "=", index*times )
    times = times+1
```

```
This is the 2 times tables.
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
2 x 11 = 22
2 x 12 = 24
>>>
```

- Turns

```
1  turns=0
2  x=3
3  while turns <22:
4    print(x)
5    x=x*3
6    print(turns)
7    turns=turns+3
8
9
```

```
[GCC 4.8.
3
0
9
3
27
6
81
9
243
12
729
15
2187
18
6561
21
```

- Number

```
number_1 = 3
number_2 = 2
total = 0

for x in range(1,5):
    number_1 = number_1*x
    number_2 = number_2 *(x+1)
    total = number_1 + number_2

print(total)
```

```
>>>
 RES
ratio
0
7
18
66
>>>
```