EXPLORE AI ACADEMY

Programmatic thinking

Exercise: Advanced if statements

Programmatic thinking

Exercise context

United Nations
Sustainable Development Goal 6

Clean water and sanitation
Ensure availability and sustainable management of water and sanitation for all.

Contamination of drinking water is a significant cause of various diseases which greatly impact human health.

The majority of these drinking water-related health problems are the result of microbial contamination. However, chemical contamination may also cause serious health concerns.

We want to create a program to monitor and improve access to clean water in rural communities.

The program needs to consider the presence and concentration of certain microbial and chemical contaminants in water sources to determine whether action is required to ensure that it is a safe drinking water source.

Exercise overview

General safety standards

Create the flowchart and pseudocode that represents the following general standards on *drinking water quality.*

We consider the presence of two microbial contaminants and the concentration limits of two chemical contaminants to determine whether action is required to ensure that the water source is safe for human consumption.

Instructions | Flowchart solution | Pseudocode solution

Presence of E. coli

Create the flowchart and pseudocode that represents the *assessment of the priority of the required action based on the presence of E. coli.*

We consider an E. coli classification and a sanitary inspection score called the ROC (Risk-Of-Contamination) score to determine the level of risk posed by the combination of the scores, and thus, the resulting priority of the action required.

Instructions | Pseudocode solution | Flowchart solution

Programmatic thinking

General safety standards

Create the flowchart and pseudocode that represents the following general standards on *drinking water quality.*

We consider two microbial and two chemical contaminants to determine whether action is required to ensure that the water source is safe for human consumption. The acceptable concentration limits are listed below.

Contaminant	Unit	Concentration limit*
Coliforms	None	Absent
E. coli	None	Absent
pH	None	6.5 to 8.5
Ammonia	mg/l	0.5

We can rewrite the contents of the table:

If coliforms or E. coli are present or the pH is lower than 6.5 or higher than 8.5 or ammonia is greater than 0.5 mg/l, then action is required.

Breaking it down further:

Action is required when:

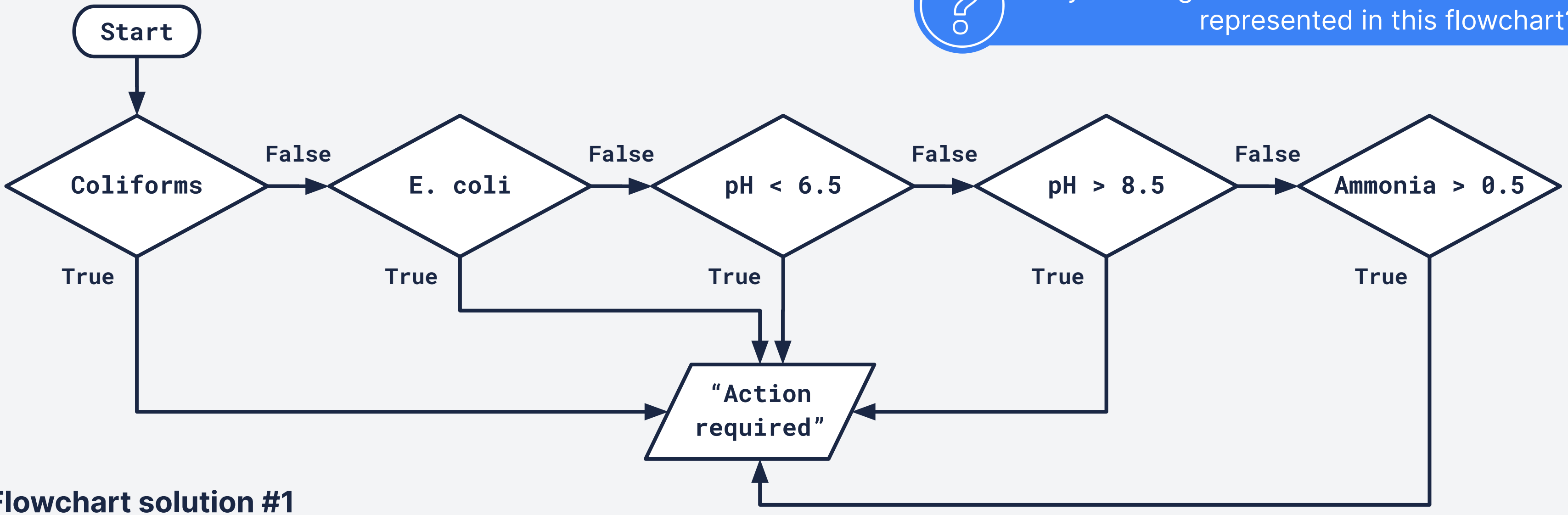
```
1. Coliforms == True OR
2. E. coli == True OR
3. pH < 6.5 OR
4. pH > 8.5 OR
5. Ammonia > 0.5
```

*Based on Kenya's Water Service Regulatory Board (WASREB) guidelines on drinking water quality. We assume the concentration limits are inclusive.

General safety standards

If any one of the contaminants is present or above the recommended concentration limits, the output will be "Action required" – this represents the OR logic in our flow of control.

Do you recognise the conditional statement represented in this flowchart?



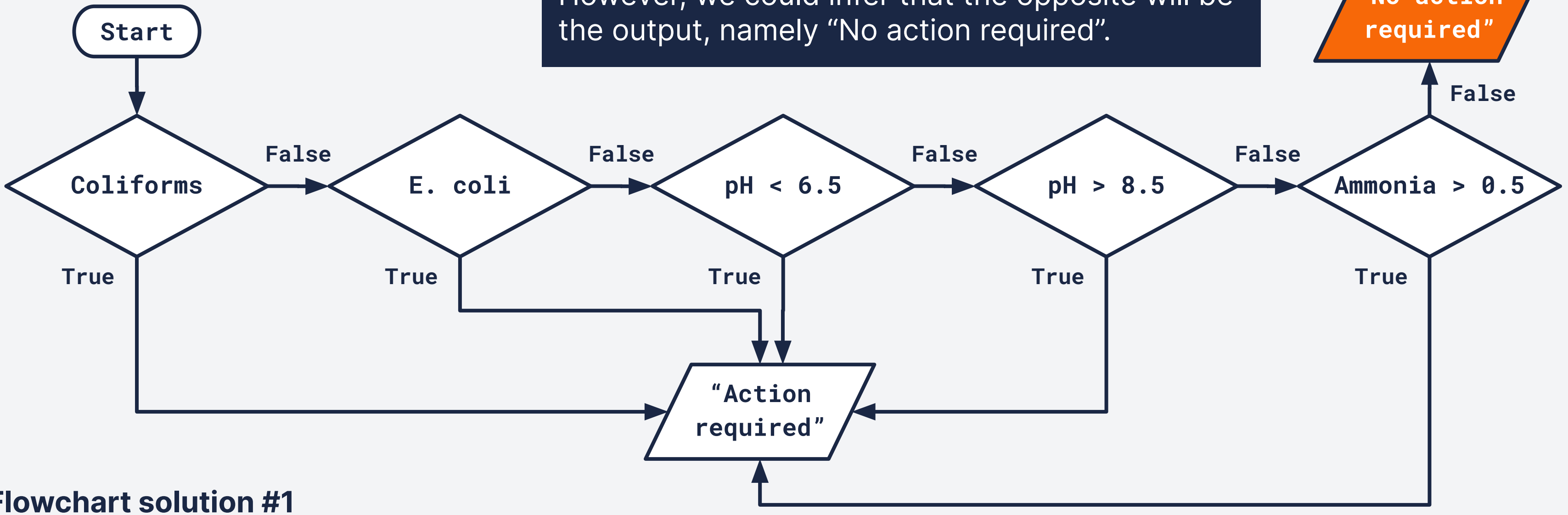
Flowchart solution #1

Programmatic thinking

General safety standards

Since the statement didn't specify that we have to output anything when none of the conditions is met, we don't have to include the False output on the last condition.

However, we could infer that the opposite will be the output, namely "No action required".



Flowchart solution #1

Programmatic thinking

General safety standards

Pseudocode solution #1

```
Start
If Coliforms then
- Output = “Action required”
Else if E. coli then
- Output = “Action required”
Else if pH < 6.5 then
- Output = “Action required”
Else if pH > 8.5 then
- Output = “Action required”
Else if Ammonia > 0.5 then
- Output = “Action required”
End if
End
```

Since the if statement already assumes we’re checking for a True result, it’s more concise to write the condition without the boolean expression.

We can add the alternative output, “No action required” by adding the following before we end the if statement:

```
Else
- Output = “No action required”
End if
```

Since the outcome of any one of the conditions is the same, we can also combine all the conditions using the OR boolean operator:

```
If Coliforms OR E. Coli OR pH < 6.5 OR pH > 8.5 OR Ammonia > 0.5 then
- Output = “Action required”
End if
```

Programmatic thinking

Presence of E. coli

Create the flowchart and pseudocode that represents the *assessment of the priority of the required action based on the presence of E. coli.*

In most cases, we would like to know **how to prioritise the required action** based on the concentration limit or classification of a specific contaminant.

We consider an E. coli classification and a sanitary inspection score called the ROC (Risk-Of-Contamination) score to determine the level of risk posed by the combination of the scores, and thus, the resulting priority of the action required.

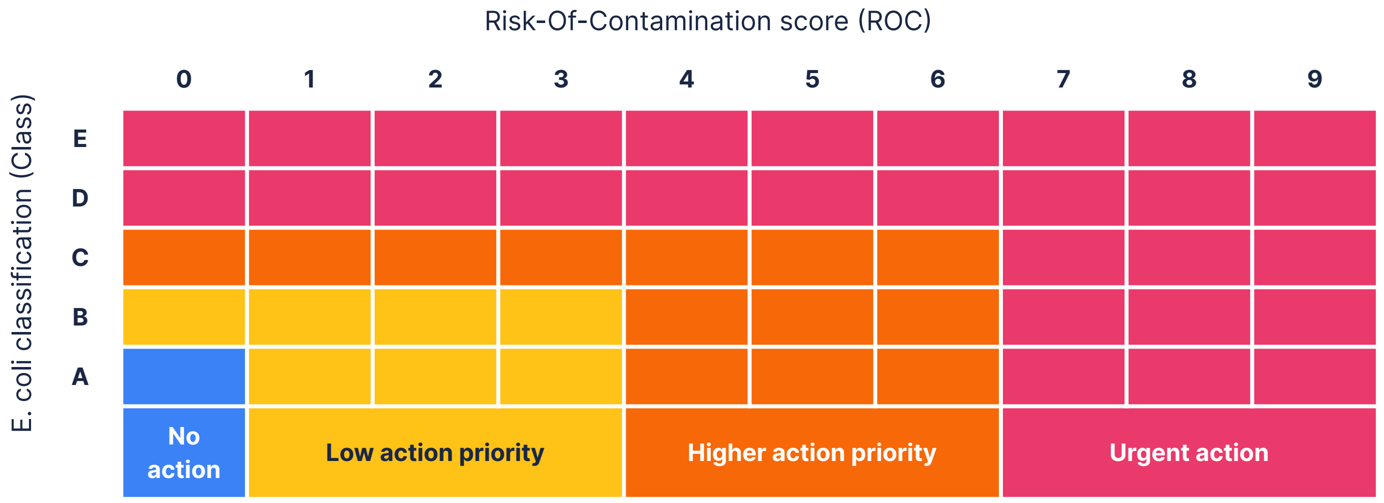
The output of the flowchart and pseudocode is one of four priorities:

- **No action**
- **Low action priority**
- **Higher action priority**
- **Urgent action**

Presence of E. coli

To set the appropriate conditions, we will need to group the outcomes based on the E. coli classification (A to E) and the ROC score (0 to 9) based on the table* below.

For example, we can already see that when the classification is D or E, the output is *very high risk: urgent action,* regardless of the ROC score.



Risk-Of-Contamination score (ROC)

E. coli classification (Class)	0	1	2	3	4	5	6	7	8	9
E	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
D	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
C	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥
B	🟨	🟨	🟨	🟨	🟧	🟧	🟧	🟥	🟥	🟥
A	🟦	🟨	🟨	🟨	🟧	🟧	🟧	🟥	🟥	🟥
	🟦 No action	🟨 Low action priority			🟧 Higher action priority			🟥 Urgent action		

*Lloyd, B.J. and Bartram, J.K., 1991. *Surveillance solutions to microbiological problems in water quality control in developing countries.* Water Science and Technology, 24(2), pp.61-75.

Presence of E. coli

Write down all the conditions for each of the four outputs.

Note how we use AND boolean operators to consider the combined requirements of the class and ROC. We use the OR boolean operators to account for the different combinations based on each class.

No action

```
(Class == A AND ROC == 0)
```

Low action priority

```
(Class == A AND ROC > 0 AND ROC < 4)
OR (Class == B AND ROC >= 0 AND ROC < 4)
```

Higher action priority

```
(Class == A AND ROC > 3 AND ROC < 7)
OR (Class == B AND ROC > 3 AND ROC < 7)
OR (Class == C AND ROC >= 0 AND ROC < 7)
```

Urgent action

```
(Class == A AND ROC > 6 AND ROC <= 9)
OR (Class == B AND ROC > 6 AND ROC <= 9)
OR (Class == C AND ROC > 6 AND ROC <= 9)
OR (Class == D AND ROC >= 0 AND ROC <= 9)
OR (Class == E AND ROC >= 0 AND ROC <= 9)
```

Presence of E. coli

Write down all the conditions for each of the four outputs.

Since we know that the ROC ranges from 0 to 9 (inclusive) we don't have to explicitly consider these bounds.

For example, if we say all ROCs smaller than 4 then we assume that it includes all ROCs that are greater than or equal to zero.

No action

```
(Class == A AND ROC == 0)
```

Low action priority

(Class == A AND ROC > 0 AND ROC < 4)
OR (Class == B ~~AND ROC >= 0~~ AND ROC < 4)

Higher action priority

```
(Class == A AND ROC > 3 AND ROC < 7)
OR (Class == B AND ROC > 3 AND ROC < 7)
```
OR (Class == C ~~AND ROC >= 0~~ AND ROC < 7)

Urgent action

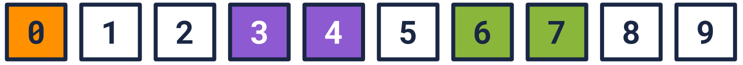
(Class == A AND ROC > 6 ~~AND ROC <= 9~~)
OR (Class == B AND ROC > 6 ~~AND ROC <= 9~~)
OR (Class == C AND ROC > 6 ~~AND ROC <= 9~~)
OR (Class == D ~~AND ROC >= 0 AND ROC <= 9~~)
OR (Class == E ~~AND ROC >= 0 AND ROC <= 9~~)

Programmatic thinking

Presence of E. coli

Write down all the conditions for each of the four outputs.

We also see that specific ROC scores indicate a divide between the priorities.



No action

```
(Class == A AND ROC == 0)
```

Low action priority

```
(Class == A AND ROC > 0 AND ROC < 4)
OR (Class == B AND ROC < 4)
```

Higher action priority

```
(Class == A AND ROC > 3 AND ROC < 7)
OR (Class == B AND ROC > 3 AND ROC < 7)
OR (Class == C AND ROC < 7)
```

Urgent action

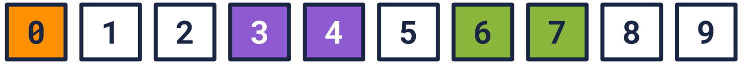
```
(Class == A AND ROC > 6)
OR (Class == B AND ROC > 6)
OR (Class == C AND ROC > 6)
OR (Class == D)
OR (Class == E)
```

Programmatic thinking

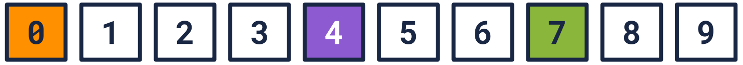
Presence of E. coli

Write down all the conditions for each of the four outputs.

We also see that specific ROC scores indicate a divide between the priorities.



However, we can also represent `> 3` as `>= 4`, and similarly `> 6` as `>= 7`. We can use three ROC scores as decisions, rather than five.



No action

```
(Class == A AND ROC == 0)
```

Low action priority

```
(Class == A AND ROC > 0 AND ROC < 4)
OR (Class == B AND ROC < 4)
```

Higher action priority

`ROC >= 4`

```
(Class == A AND ROC > 3 AND ROC < 7)
OR (Class == B AND ROC > 3 AND ROC < 7)
OR (Class == C AND ROC < 7)
```

Urgent action

`ROC >= 7`

```
(Class == A AND ROC > 6)
OR (Class == B AND ROC > 6)
OR (Class == C AND ROC > 6)
OR (Class == D)
OR (Class == E)
```

Programmatic thinking

Presence of E. coli

Write down all the conditions for each of the four outputs.

We can write down an if-else-if ladder where the conditions are only related to the classification (Class).

```
If Class == A then
- what needs to happen if the condition is true
Else if Class == B then
- what needs to happen if the condition is true
Else if Class == C then
- what needs to happen if the condition is true
Else
- what needs to happen if none of the conditions is true
  (i.e. Class == D OR Class == E)
End if
```

No action

```
(Class == A AND ROC == 0)
```

Low action priority

```
(Class == A AND ROC > 0 AND ROC < 4)
OR (Class == B AND ROC < 4)
```

Higher action priority

```
(Class == A AND ROC >= 4 AND ROC < 7)
OR (Class == B AND ROC >= 4 AND ROC < 7)
OR (Class == C AND ROC < 7)
```

Urgent action

```
(Class == A AND ROC >= 7)
OR (Class == B AND ROC >= 7)
OR (Class == C AND ROC >= 7)
OR (Class == D)
OR (Class == E)
```

Programmatic thinking

Presence of E. coli

Write down all the conditions for each of the four outputs.

Let's only consider the ROC conditions and related output when **`Class == A`**.

```
If Class == A then
- If ROC == 0 then
- - Output = “No action”
- Else if ROC < 4 then
- - Output = “Low action priority”
- Else if ROC < 7 then
- - Output = “Higher action priority”
- Else
- - Output = “Urgent action”
- End if
Else if Class == B then
- ...
```

No action

`(Class == A AND ROC == 0)`

Low action priority

`(Class == A AND ROC > 0 AND ROC < 4)`

Higher action priority

`(Class == A AND ROC >= 4 AND ROC < 7)`

Urgent action

`(Class == A AND ROC >= 7)`

Note, we don't need to test for **`ROC > 0`** because when ROC != 0 in the first condition then ROC can only be greater than zero when the program gets to **`ROC < 4`**.

Programmatic thinking

Presence of E. coli

Write down all the conditions for each of the four outputs.

Let’s only consider the ROC conditions and related output when **`Class == B`**.

```
- ...
Else if Class == B then
- If ROC < 4 then
- - Output = “Low action priority”
- Else if ROC < 7 then
- - Output = “Higher action priority”
- Else
- - Output = “Urgent action”
- End if
Else if Class == C then
- ...
```

No action

Low action priority

`(Class == B AND ROC < 4)`

Higher action priority

`(Class == B AND ROC >= 4 AND ROC < 7)`

Urgent action

`(Class == B AND ROC >= 7)`

Note, we also don’t need to test for **`ROC >= 4`** because when ROC is not smaller than 4 in the first condition then ROC can only be greater than four when the program gets to **`ROC < 7`**.

Programmatic thinking

Presence of E. coli

Write down all the conditions for each of the four outputs.

Let’s only consider the ROC conditions and related output when **`Class == C`**.

```
- ...
Else if Class == C then
- If ROC < 7 then
- - Output = “Higher action priority”
- Else
- - Output = “Urgent action”
- End if
Else
- ...
```

No action

Low action priority

Higher action priority
(Class == C AND ROC < 7)

Urgent action
(Class == C AND ROC >= 7)

Note, we also don’t have to specifically state the last condition (**ROC >= 7**) because when ROC is not smaller than 7 it can only be greater than or equal to 7. We simply use an else statement.

Presence of E. coli

Write down all the conditions for each of the four outputs.

Let's only consider the ROC conditions and related output when `Class == D` or `Class == E`.

```
- ...
Else
- Output = "Urgent action"
End if
```

No action

Low action priority

Higher action priority

Urgent action
`(Class == D)`
`(Class == E)`

When the class is neither A, B, nor C, then the flow of control reverts to the else of the if-else-if ladder. Since the output is "Urgent action" when **`Class == D`** or **`Class == E`** for any ROC, we do not require a nested if statement.

Programmatic thinking

Presence of E. coli

Write down all the conditions for each of the four outputs.

Consider the below pseudocode for the end of our flow of control:

```
- ...
Else if Class == C then
- If ROC < 7 then
- - Output = Higher action priority
- Else
- - Output = Urgent action
- End if
Else
- Output = Urgent action
End if
```

We see that when the class is NOT A nor B, the output can only be “Higher action priority” or “Urgent action”.

Furthermore, when the class is NOT C AND ROC is greater than or equal to 7 (NOT smaller than 7) the output can only be “Urgent action”.

In other words, we can combine the two conditions with the AND boolean operator to test both conditions at the same time.

```
- ...
Else if Class == C AND ROC < 7 then
- Output = Higher action priority
Else
- Output = Urgent action
End if
```

Programmatic thinking

Presence of E. coli

Initial pseudocode solution #0

```
If Class == A then
- If ROC == 0 then
- - Output = "No action"
- Else if ROC < 4 then
- - Output = "Low action priority"
- Else if ROC < 7 then
- - Output = "Higher action priority"
- Else
- - Output = "Urgent action"
- End if
Else if Class == B then
- If ROC < 4 then
- - Output = "Low action priority"
- Else if ROC < 7 then
- - Output = "Higher action priority"
- Else
- - Output = "Urgent action"
- End if
Else if Class == C AND ROC < 7 then
- Output = "Higher action priority"
Else
- Output = "Urgent action"
End if
```

Combining the various parts we've constructed per class, we notice that, except for when ROC == 0, Class A and B are the same.

```
...
Else if Class == A OR Class == B then
- If ROC < 4 then
- - Output = "Low action priority"
- Else if ROC < 7 then
- - Output = "Higher action priority"
- Else
- - Output = "Urgent action"
- End if
...
```

We use the **OR boolean operator** to combine the same conditions together.

However, we need to **account for the exception**, i.e. when ROC == 0, to set the output to "No action".

Presence of E. coli

Alternative pseudocode solution #1

```
If Class == A AND ROC == 0 then
- Output = "No action"
Else if Class == A OR Class == B then
- If ROC < 4 then
- - Output = "Low action priority"
- Else if ROC < 7 then
- - Output = "Higher action priority"
- Else
- - Output = "Urgent action"
- End if
Else if Class == C AND ROC < 7 then
- Output = "Higher action priority"
Else
- Output = "Urgent action"
End if
```

We account for the exception by adding a **preceding condition** that simultaneously accounts for both the class and ROC conditions using the **AND boolean operator**.

Note, the `(Class == A AND ROC == 0)` condition **must precede** the `(Class == A OR Class == B)` condition.

Consider the following scenario:

Input **Class** as **A** and **ROC** as **0** and compare the outputs of `(Class == A AND ROC == 0)` and `(Class == A OR Class == B)`. Are any of the conditions true for this input?

Programmatic thinking

Presence of E. coli

We can identify other similarities and exceptions:

- Only when `ROC < 4` and the class is either **A** or **B**, is the output "Low action priority".
- When the class is **A**, **B**, or **C**, `ROC >= 7` indicates an output of "Urgent action". However, when the class is **D** or **E**, regardless of the ROC, the output is also "Urgent action".

Alternative pseudocode solution #2

```
If Class == A AND ROC == 0 then
- Output = "No action"
Else if (Class == A OR Class == B) AND ROC < 4 then
- Output = "Low action priority"
Else if Class == D OR Class == E OR ROC >= 7 then
- Output = "Urgent action"
Else
- Output = "Higher action priority"
End if
```

There are many different ways to find the same output to the same problem using pseudocode:

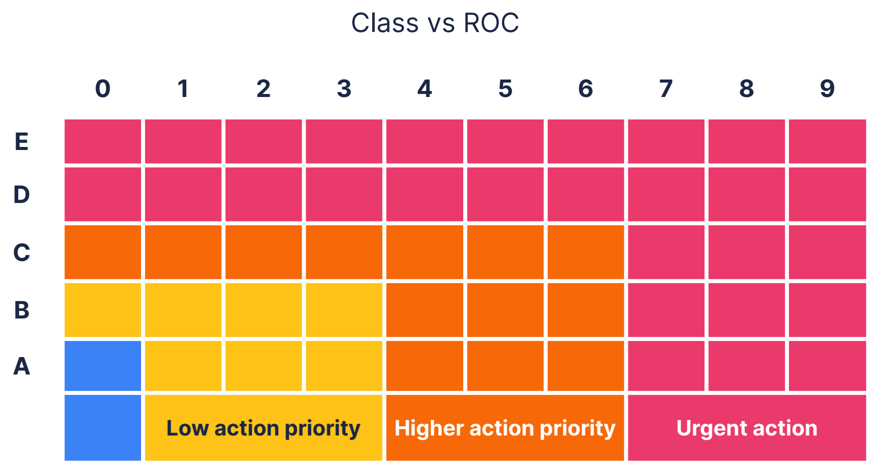
- Grouping by ROC for the if-else-if ladder in either ascending or descending order.
- Mixing ROC and class conditions in the if-else-if ladder.
- Setting conditions for classes in the if-else-if ladder from E to A rather than A to E.
- Using nested if statements for a shorted if-else-if ladder.

Shorter and more efficient pseudocode often leads to better and more efficient implementations of the solution when coding in a specific programming language.

Programmatic thinking

Presence of E. coli

We’ve already mapped out the logic of control required while writing the pseudocode. See if you can use these groupings to create a **flowchart** that includes only one instance of each output.



No action

```
(Class == A AND ROC == 0)
```

Low action priority

```
(Class == A AND ROC > 0 AND ROC < 4)
OR (Class == B AND ROC < 4)
```

Higher action priority

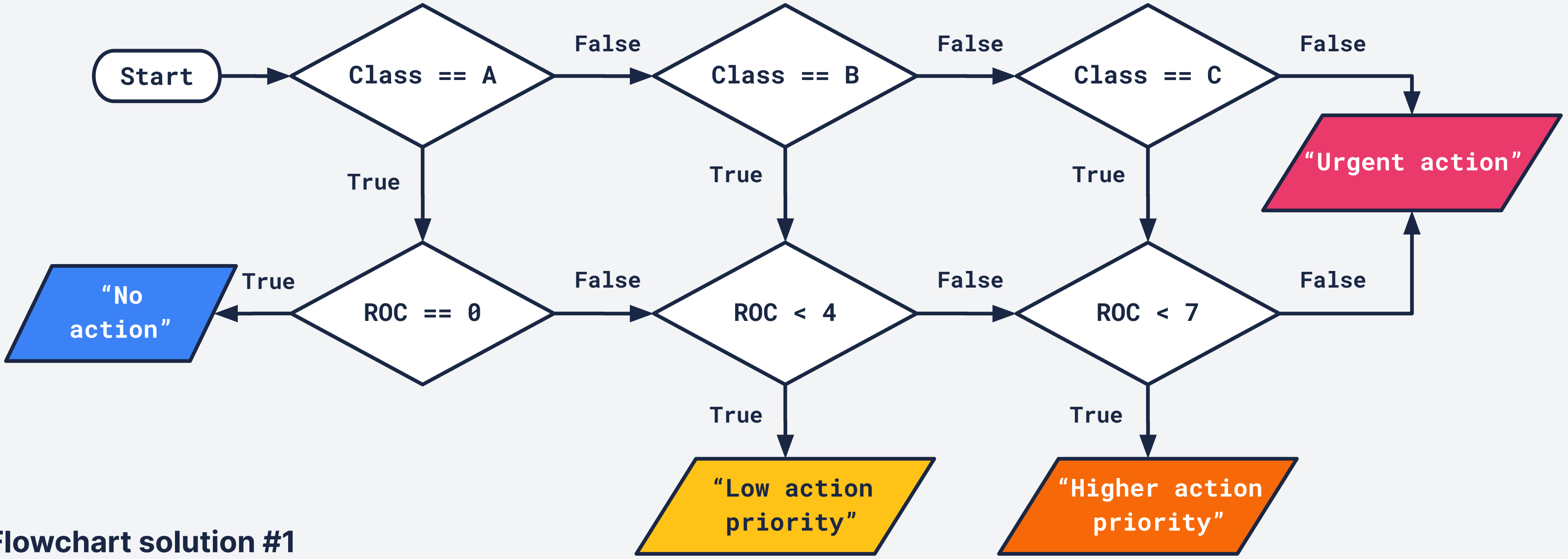
```
(Class == A AND ROC >= 4 AND ROC < 7)
OR (Class == B AND ROC >= 4 AND ROC < 7)
OR (Class == C AND ROC < 7)
```

Urgent action

```
(Class == A AND ROC >= 7)
OR (Class == B AND ROC >= 7)
OR (Class == C AND ROC >= 7)
OR (Class == D)
OR (Class == E)
```

Programmatic thinking

Presence of E. coli



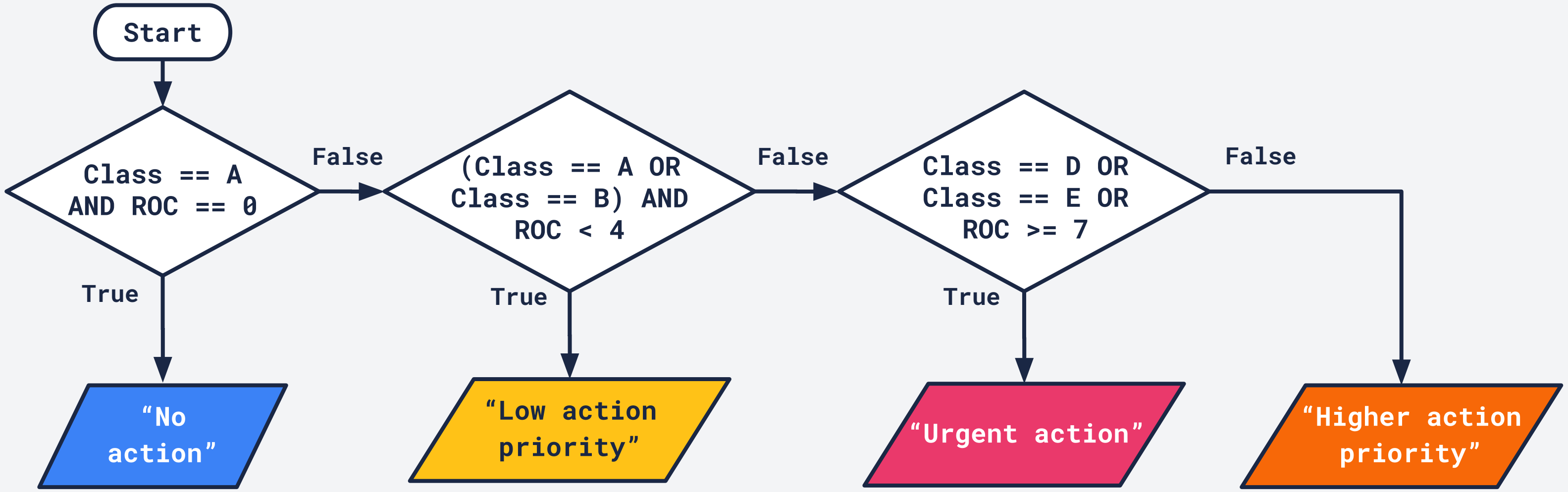
Flowchart solution #1

Write down the pseudocode of this flowchart using both if-else-if ladders and nested if statements. Does the pseudocode seem more or less efficient than our last pseudocode iteration?

Programmatic thinking

Presence of E. coli

Flowchart to the alternative pseudocode solution #2



Although our second pseudocode solution was the most efficient, the flowchart of this pseudocode is unintuitive. This shows that the most efficient pseudocode doesn’t necessarily translate into the most intuitive flowchart, and vice versa.