IG Computer Science

Unit 1: Problem Solving

Part 3: Pseudocode

Topic 1: Understanding Algorithms
Topic 2: Creating Algorithms

Lecture Contents

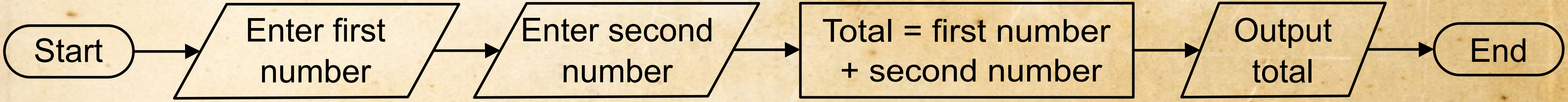
- Vocabulary
- Naming Conventions
- Pseudocode
- Readings:
  - Text under PSEUDOCODE heading on pages 7-9
  - Example of pseudocode, Activity 8, page 13-14

Vocabulary – variable

- Computer programs need to store data
- The ***value*** of some such data may change as the program runs
- Programs refer to the place this data is stored as a ***variable***

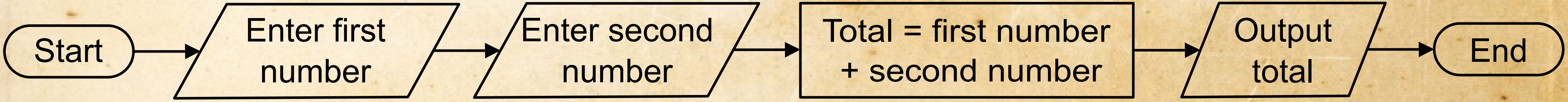
Vocabulary – ***variable***

- Computer programs need to store data
- The ***value*** of some such data may change as the program runs
- Programs refer to the place this data is stored as a ***variable***



Vocabulary – *variable*

- Computer programs need to store data
- The ***value*** of some such data may change as the program runs
- Programs refer to the place this data is stored as a ***variable***



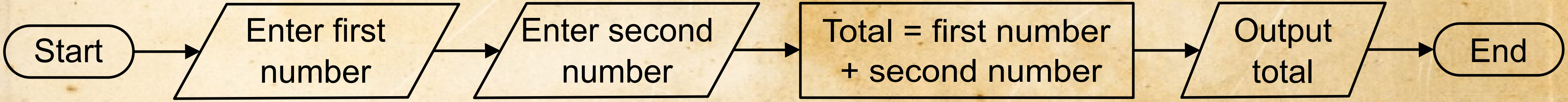
- In the example algorithm from above, the program needs a place to store the “first number” and the “second number”.
  - So this algorithm needs at least two ***variables*** (perhaps another *variable* for the “Total”).

Vocabulary – ***identifier*** or ***label***

- Programers need a way to refer to its *variables*.
- A unique name given to a *variable* is called an ***identifier*** or a ***label***.

Vocabulary – ***identifier*** or ***label***

- Programers need a way to refer to its *variables*.
- A unique name given to a *variable* is called an ***identifier*** or a ***label***.



Start

Enter first number

Enter second number

Total = first number + second number

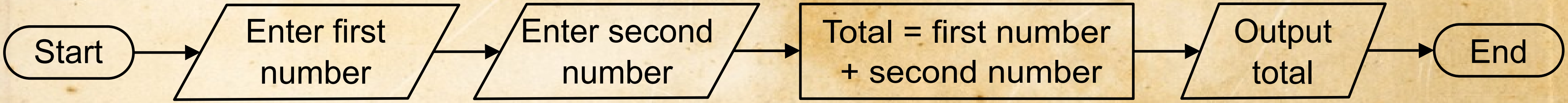
Output total

End

- In the example above, we might label the “first number” as `firstNumber` and the second as `secondNumber`.

Vocabulary – ***identifier*** or ***label***

- Programers need a way to refer to its *variables*.
- A unique name given to a *variable* is called an ***identifier*** or a ***label***.



- In the example above, we might label the "first number" as `firstNumber` and the second as `secondNumber`.
- Basically every programming language does not allow any whitespace within an ***identifier***.
  - For example, we cannot use `first number` as a label because there is a space between the two words.

Vocabulary – ***pseudocode*** (为代码)

- ***Pseudocode*** is structured, code-like language that can be used to describe an algorithm.
  - There isn’t really a standard form for pseudocode, just any writing that is organized and looks like code, but is not actually code
  - Unfortunately, for Pearson IG Computer Science, you will need to learn their version of pseudocode.

Vocabulary

- Vocabulary
  - ***variable***: a “container” used to store data. The value of the data may change as the program is run.
  - ***identifier*** or ***label***: the unique name given to a variable
    - or the unique name given to a constant, function, class, etc., (which we will learn about later)
  - ***pseudocode***: structured, code-like language that can be used to describe an algorithm

Naming Conventions

- Variable names should be **descriptive** to make the code easy to read.
- Since we cannot use whitespace, lower case may be hard to read.
  - hardtoreadthisvariblename

Naming Conventions

- Variable names should be **descriptive** to make the code easy to read.
- Since we cannot use whitespace, lower case may be hard to read.
  - `hardtoreadthisvariblename`
- So, programmers have come up with conventions to make *identifiers* easier to read.
  - `easierToReadThisVariableName`
  - `this_one_is_easy_to_read_too`
  - `maybe-you-like-this-style`
- Each programming language has its preferred convention.

Naming Conventions

- The next slides will give an introduction different styles for naming conventions.
- The names of these conventions will not be tested
- You must use the appropriate naming conventions when you write your code!

Naming Conventions – *Camel Case*

- ***Camel case***, sometimes called *lower camel case*:
  - The first letter of the first word is lower case
  - The first letter of every subsequent word is upper case
  - All other letters are lower case
  - Examples:
    - `firstNumber`
    - `userName`
    - `studentDateOfBirth`

Java uses camel case for names of *variables* and *methods*.
Python does not use lower camel case.
C# uses camel case for names of *variables*

Naming Conventions – *Pascal Case*

- ***Pascal case***, sometimes called *upper camel case*:
  - The first letter of every word is upper case
  - All other letters are lower case
  - Examples:
    - `FirstNumber`
    - `UserName`
    - `StudentDateOfBirth`

Java uses Pascal case for *class* names.

Python uses camel case *class* names.

C# uses camel case for names of *constants*, *classes*, and *methods*, etc.

Naming Conventions – *Upper Case*

- ***Upper case***:
  - All letters are upper case
  - Usually an underscore between each word for improved readability
  - Examples:
    - `FIRST_NUMBER`
    - `USER_NAME`
    - `STUDENT_DATE_OF_BIRTH`

Java uses Pascal case for names of *constants*.
Python uses Pascal case for names of *constants*.
C# does not use upper case

Naming Conventions – *Snake Case*

- ***Snake case***:
  - All letters are lower case
  - An underscore between each word for improved readability
  - Examples:
    - `first_number`
    - `first_name`
    - `student_date_of_birth`

Java does not use snake case.

Python uses snake case for names of variables, functions, modules, etc.

C# does not use snake case

Naming Conventions – *Kebab Case*

- ***kebab case*** (kebab = 串儿):
  - All letters are lower case
  - A hyphen between each word for improved readability
  - Examples:
    - `first-number`
    - `first-name`
    - `student-date-of-birth`

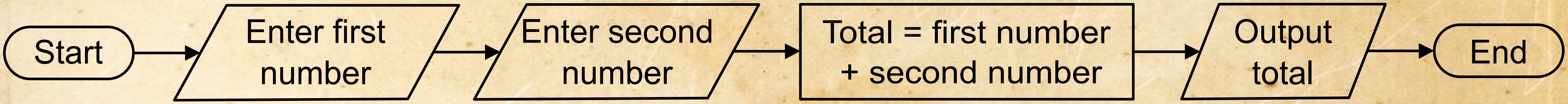
  Java does not use kebab case.
  Python and C# do not use kebab case.
  Kebab case is used in HTML for attributes

Naming Conventions - Summary

- Different styles for naming conventions
  - camelCase / lowerCamelCase
  - PascalCase / UpperCamelCase
  - UPPER_CASE
  - snake_case
  - kebab-case
- You won’t be tested on the names, but you must be aware of these, and use the appropriate one for the programming language you’re using.

Pseudocode(为代码)

- Here is an example of Pearson Edexcel’s ***pseudocode***.



Start

Enter first number

Enter second number

Total = first number + second number

Output total

End

```
SEND ‘Please enter the first number’ TO DISPLAY
RECEIVE firstNumber FROM KEYBOARD
SEND ‘Please enter the second number’ TO DISPLAY
RECEIVE secondNumber FROM KEYBOARD
SET total TO firstNumber + secondNumber
SEND total TO DISPLAY
```

Pseudocode

- Some pseudocode commands, with examples:
  - `SEND` *<expression>* `TO` *<device>*
    - `SEND ‘Have a good day.’ TO DIPLAY`
  - `RECEIVE` *<variable>* `FROM` *<device>*
    - `RECEIVE userName FROM KEYBOARD`
  - `SET` *<variable>* `TO` *<expression>*
    - `SET circumference TO radius * 2 * PI`

- Note: an *<expression>* can be a single value or a calculation.

Pseudocode

- Selection:

  -
    ```
    IF <expression> THEN
          <pseudocode commands>
    END IF
    ```

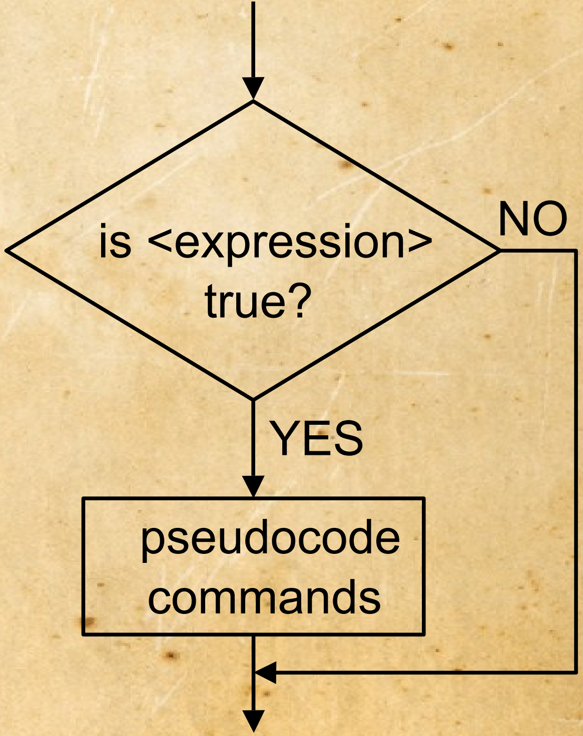
    -
      ```
      IF greet = ‘true’ THEN
          SEND ‘Good day!’ TO DISPLAY
      END IF
      ```

Pseudocode

- Selection:

  -
    ```
    IF <expression> THEN
        <pseudocode commands>
    END IF
    ```

    -
      ```
      IF greet = ‘true’ THEN
          SEND ‘Good day!’ TO DISPLAY
      END IF
      ```



Pseudocode

- Selection:

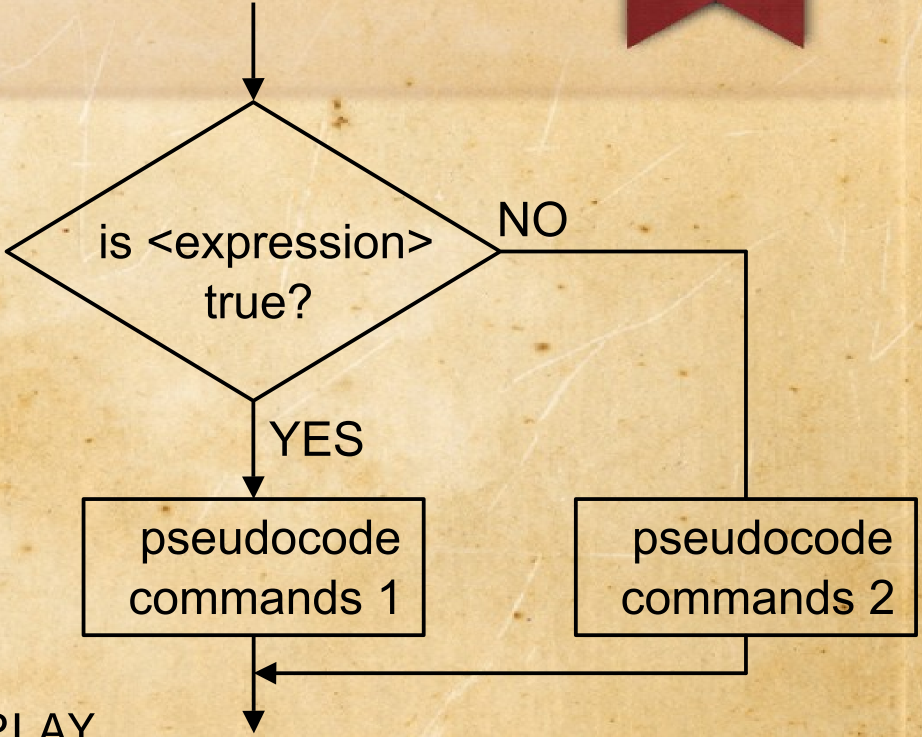
  -
    ```
    IF <expression> THEN
        <pseudocode commands 1>
    ELSE
        <pseudocode commands 2>
    END IF
    ```

    -
      ```
      IF answer = 10 THEN
          SEND ‘You win!’ TO DISPLAY
      ELSE
          SEND ‘Sorry, try again!’ TO DISPLAY
      END IF
      ```

Pseudocode

- Selection:
  -
    ```
    IF <expression> THEN
        <pseudocode commands 1>
    ELSE
        <pseudocode commands 2>
    END IF
    ```

    -
      ```
      IF answer = 10 THEN
          SEND 'You win!' TO DISPLAY
      ELSE
          SEND 'Sorry, try again!' TO DISPLAY
      END IF
      ```



Pseudocode

- For ***iteration*** (loops)

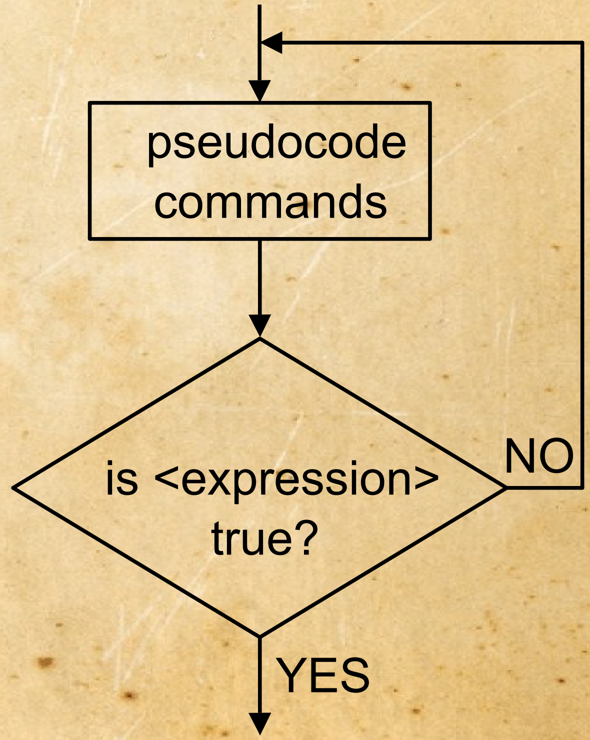
  -
    ```
    REPEAT
        <pseudocode commands>
    UNTIL <expression>
    ```

      -
        ```
        SET count TO 1
        REPEAT
            SEND count TO DISPLAY
            SET count TO count + 1
        UNTIL count = 10
        ```

Pseudocode

- For ***iteration*** (loops)
  - REPEAT
    <pseudocode commands>
    UNTIL *<expression>*

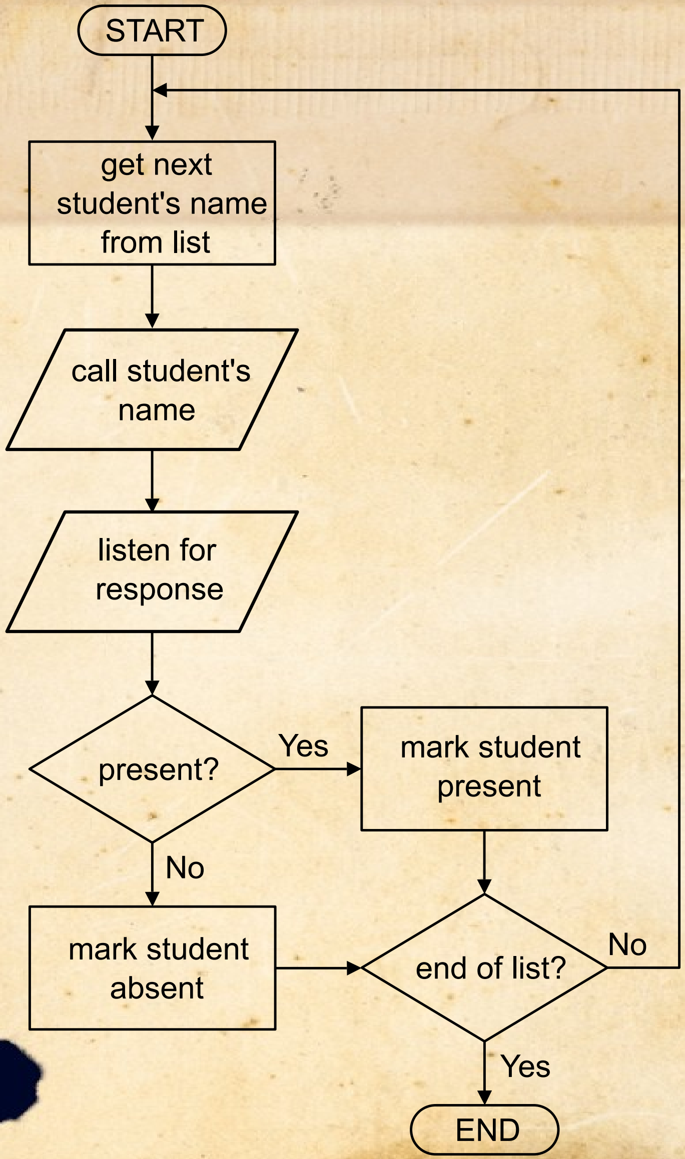
```
SET count TO 1
REPEAT
    SEND count TO DISPLAY
    SET count TO count + 1
UNTIL count = 10
```

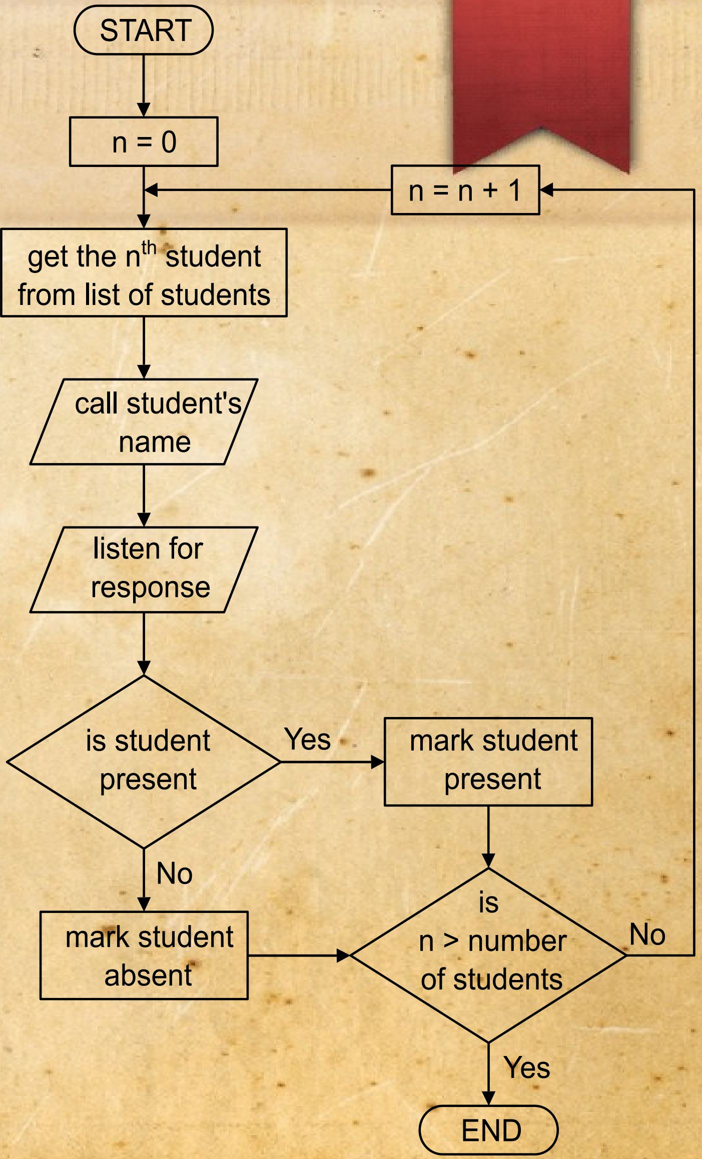


Flowchart Practice – Class Attendance

- Draw a flowchart that represents the algorithm to take class attendance.
  - Assume the teacher already has a list of students, so there is no need to take it as input.
  - Don’t worry about waiting for the student reply – assume the input block will wait the appropriate amount of time and the student will either be present or absent.
  - Make sure you ***iterate*** through the entire class list

Class Attendance

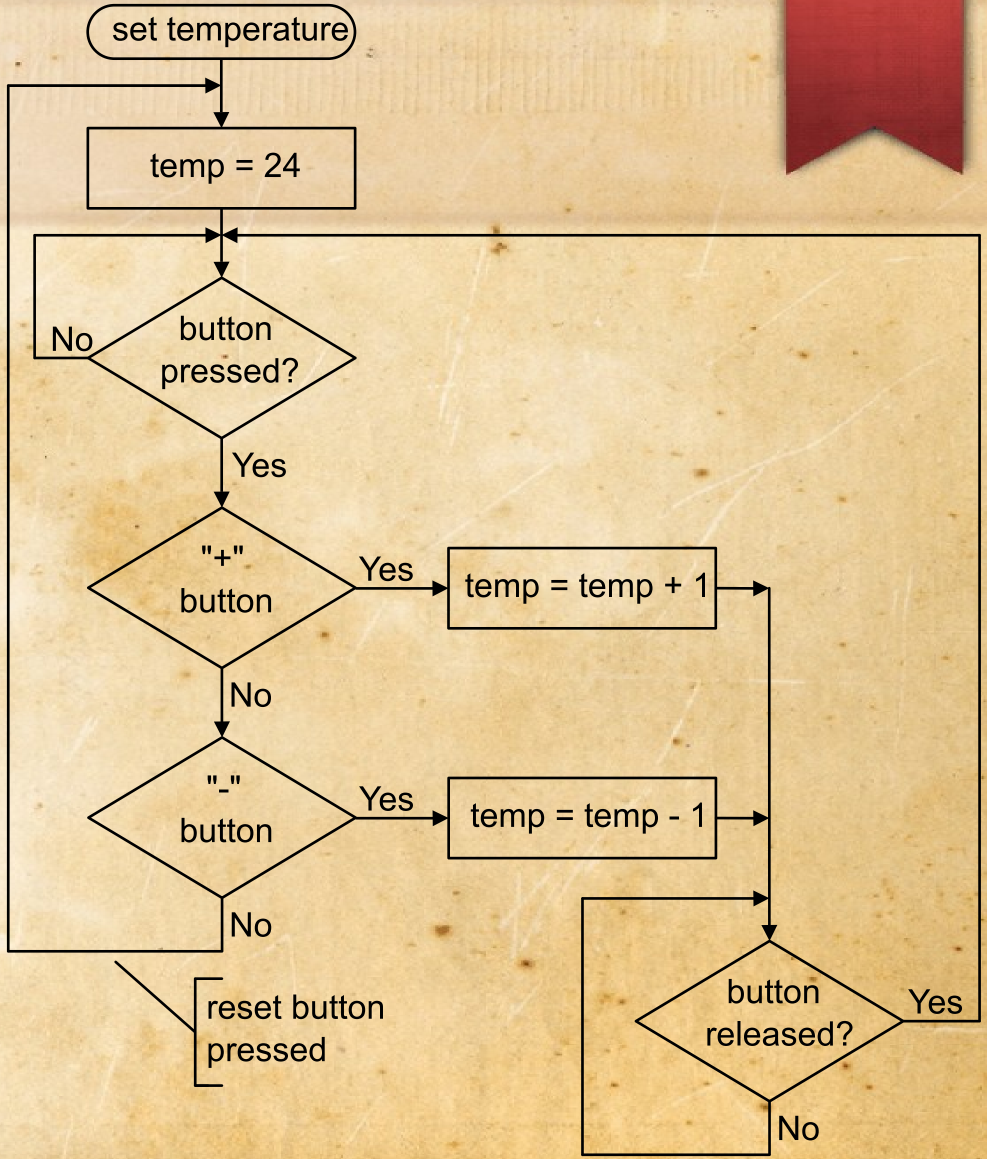




Flowchart Practice - Thermostat

- Make a flowchart that represents the following thermostat algorithm:
  - the thermostat has 3 buttons: : “+”, “-”, and “*reset*”
  - The *reset* button will reset the thermostat to 24 degrees.
  - The + button will increase the set temperature by one degree
  - The – button will decrease the set temperature by one degree

Flowchart Practice – Thermostat



IG Computer Science

Unit 1: Problem Solving

Part 3: Pseudocode

Topic 1: Understanding Algorithms
Topic 2: Creating Algorithms