2023

Review

Logic Algorithm

Tri Hadiah Muliawati

Politeknik Elektronika Negeri Surabaya
Department of Informatics and Computer Engineering,

Review

Let’s throwback to algorithm

Algorithm is a **way of specifying a multi-step task**, and is especially useful when we wish **to explain to a third party (be it human or machine) how to carry out steps with extreme precision**.

The properties of an algorithm are:

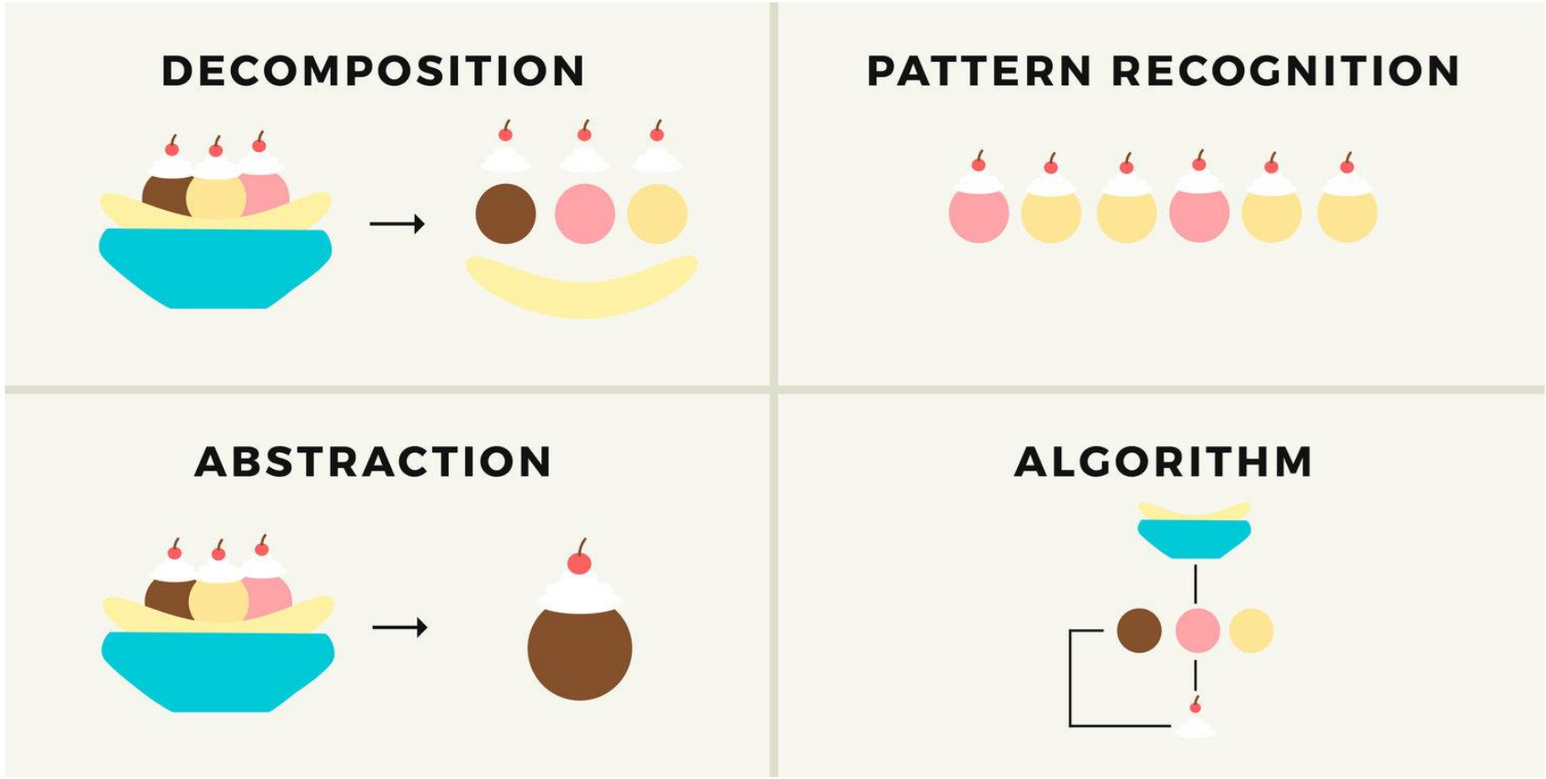
1. Finiteness
2. Definiteness
3. Input
4. Output
5. Effectiveness

Why algorithm (alone) is not sufficient to solve a problem?

Sometimes we deal with **a complex problem** which may overwhelm us and makes it **difficult to directly form an algorithm to solve it**.

Computational Thinking (CT) breaks down a complex problem into smaller, more comprehensible tasks. Therefore, CT helps us to recognize that some tasks that might **seem very difficult at first are actually very doable**.

Pillars of CT

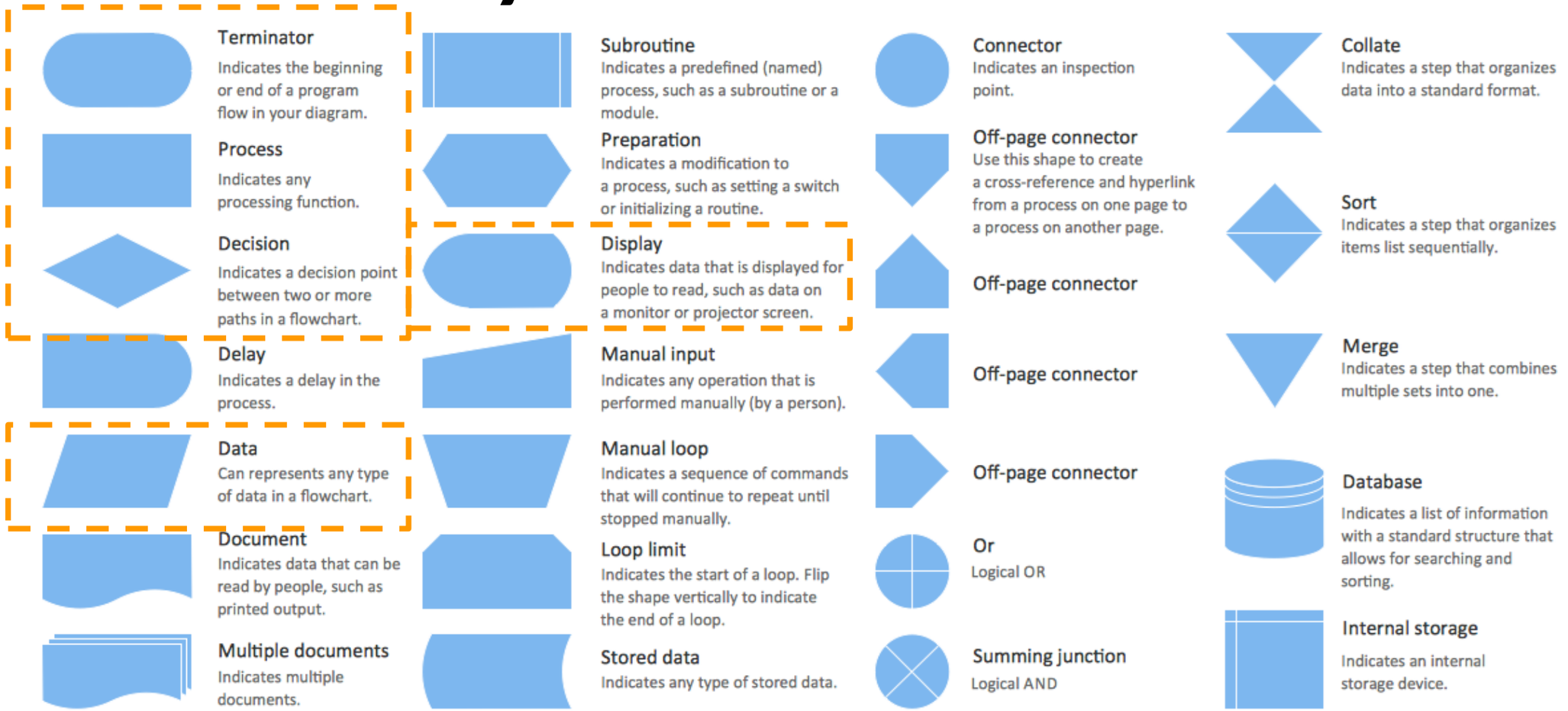


- Use them appropriately.
- It is not necessary to use all pillars when solving every problem.

Flowcharts

- Flowcharts can be seen as graphical representations of algorithms.
- Flowcharts are more intuitive and make it easy to understand the algorithms. (depending on the complexity of the problem however, sometimes algorithms may better represent the steps to solve the problem)
- We use standard symbols and shapes in a Flowchart to show the sequential steps of an algorithm.

Flowcharts Symbols



Terminator
Indicates the beginning or end of a program flow in your diagram.

Process
Indicates any processing function.

Decision
Indicates a decision point between two or more paths in a flowchart.

Delay
Indicates a delay in the process.

Data
Can represents any type of data in a flowchart.

Document
Indicates data that can be read by people, such as printed output.

Multiple documents
Indicates multiple documents.

Subroutine
Indicates a predefined (named) process, such as a subroutine or a module.

Preparation
Indicates a modification to a process, such as setting a switch or initializing a routine.

Display
Indicates data that is displayed for people to read, such as data on a monitor or projector screen.

Manual input
Indicates any operation that is performed manually (by a person).

Manual loop
Indicates a sequence of commands that will continue to repeat until stopped manually.

Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.

Stored data
Indicates any type of stored data.

Connector
Indicates an inspection point.

Off-page connector
Use this shape to create a cross-reference and hyperlink from a process on one page to a process on another page.

Off-page connector

Off-page connector

Off-page connector

Or
Logical OR

Summing junction
Logical AND

Collate
Indicates a step that organizes data into a standard format.

Sort
Indicates a step that organizes items list sequentially.

Merge
Indicates a step that combines multiple sets into one.

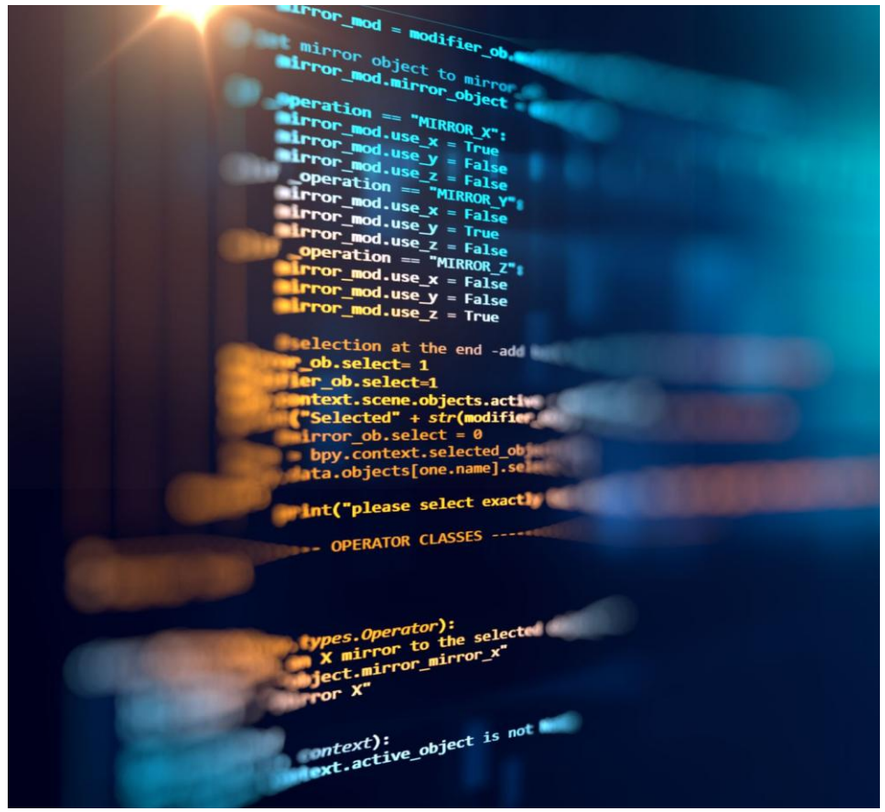
Database
Indicates a list of information with a standard structure that allows for searching and sorting.

Internal storage
Indicates an internal storage device.

Flowchart can help to visualize the algorithm, thus we can understand it better. However, it is **not easy to turn a flowchart into a practical code**.
Therefore, we can use **pseudocode** as an assisting tool **to turn algorithm and/or flowchart to code**.

Pseudocode

- Pseudocode is a fake code that is an explanation of how to **solve a problem**
- Pseudocode uses a language that **almost** like programming language
- Apart from that, pseudocode usually uses language that is **easy to understand** and also **more concise** than algorithms
- In pseudocode, there is **no standard syntax or grammar rules**
- Therefore, we can apply this pseudocode in various programming languages



Example

Write a solution to check whether a student pass “Pemrograman 1” subject or not. He / She will pass if the final score is more than or equal to 56.

Answer

Objective: Check whether a student pass “Pemrograman 1” subject or not. He / She will pass if the final score is more than or equal to 56.

Algorithm:

- Input the final score
- Check whether the final score is more than or equal to 56
- If yes, print he/she passes the subject
- If no, print he/she does not pass the subject

Pseudocode:

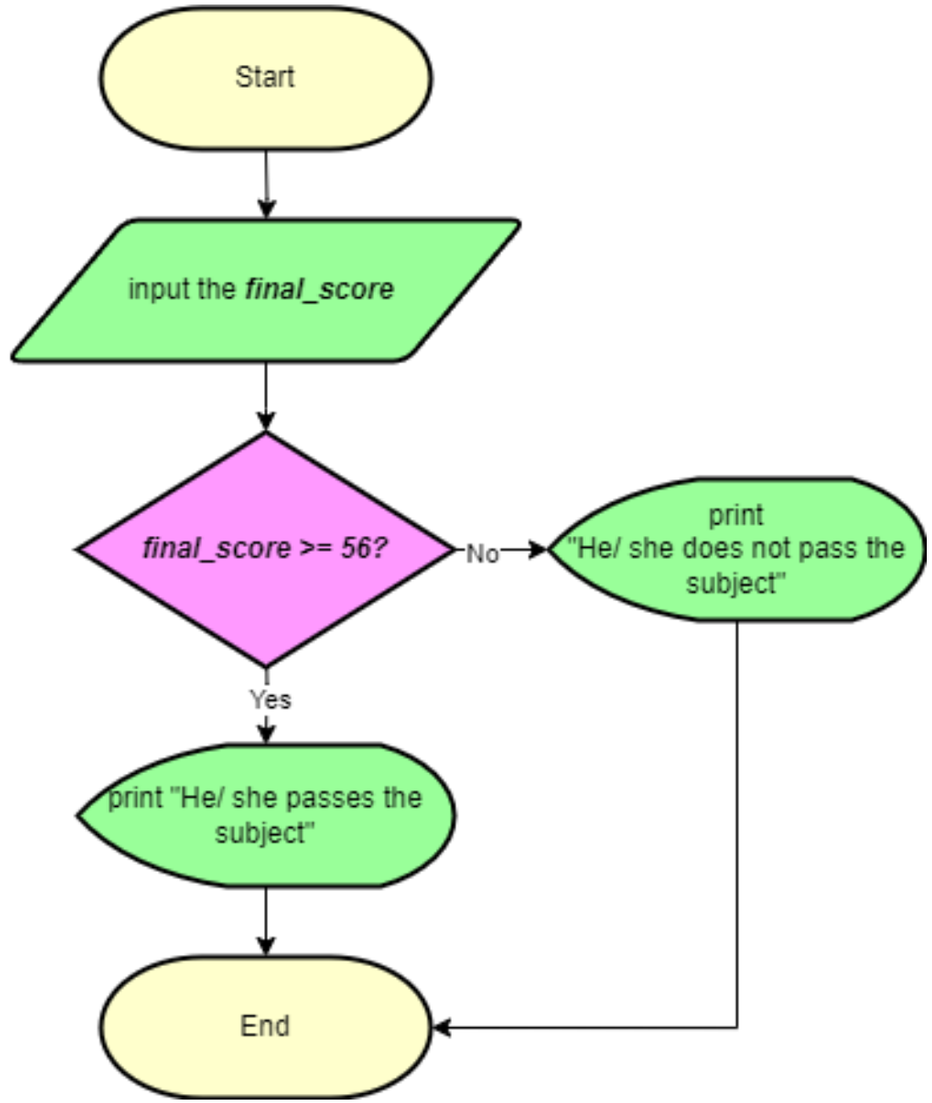
- INPUT the *final_score*
- IF *final_score* >= 56 THEN
- PRINT “he/she passes the subject”
- ELSE
- PRINT “he/she does not pass the subject”
- ENDIF

Answer

Objective: Check whether a student pass “Pemrograman 1” subject or not. He / She will pass if the final score is more than or equal to 56.

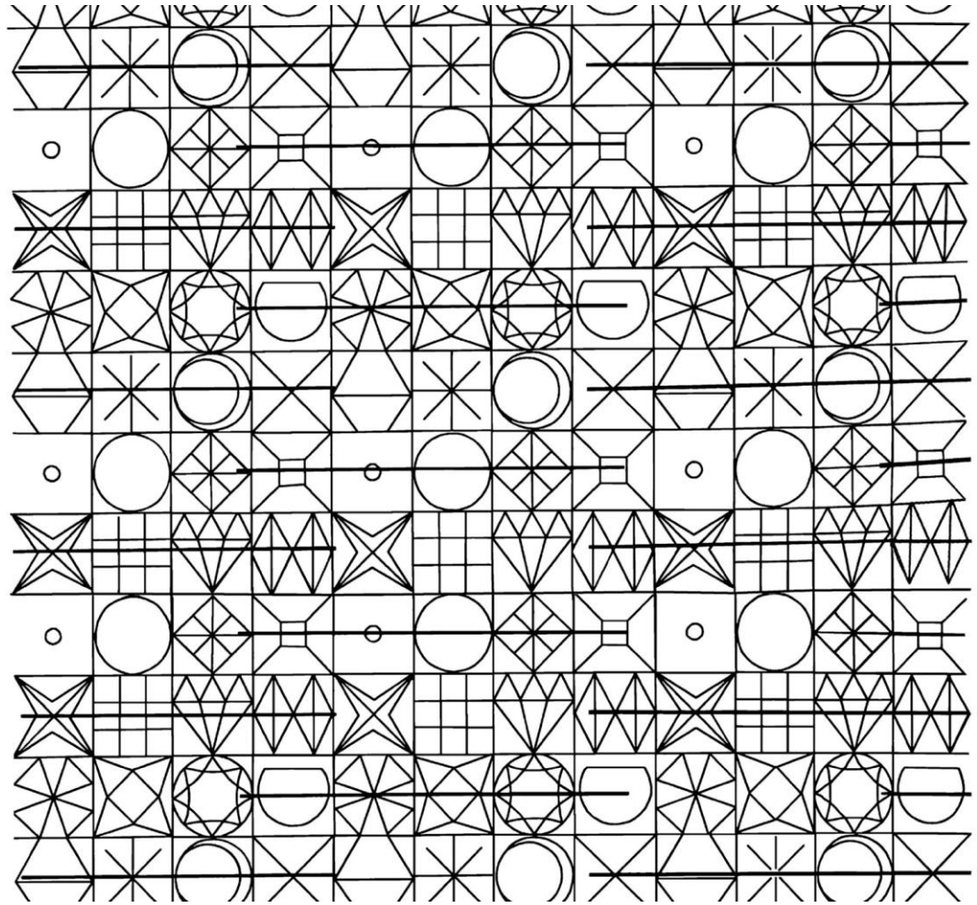
Pseudocode:

- INPUT the *final_score*
- IF *final_score* >= 56 THEN
- PRINT “he/she passes the subject”
- ELSE
- PRINT “he/she does not pass the subject”
- ENDIF



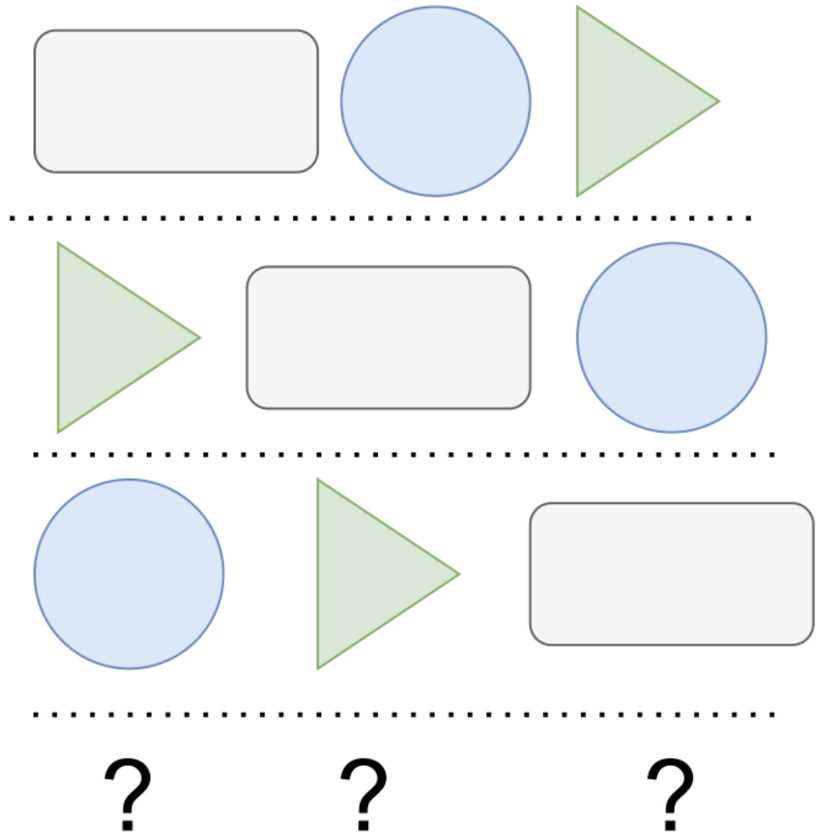
Pattern Recognition

- Pattern recognition is the **skill of recognizing the similarities and differences** between concepts and objects.
- Pattern recognition is the **ability to analyze and identify the shared characteristics** between parts of a decomposed problem.
- Pattern recognition is a core computational thinking skill that **helps in creating shortcuts to solve complex problems**.
- Pattern recognition helps **avoid duplications**, and **not to reinvent the wheel**!



Example

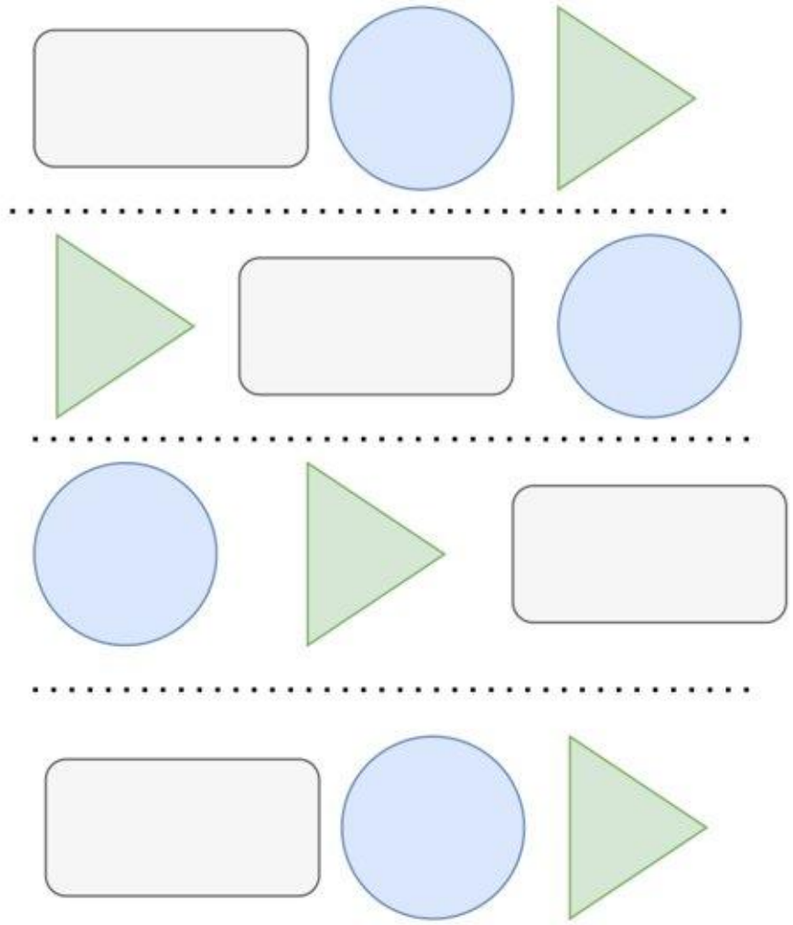
Can you identify a pattern
in this picture and predict
the arrangement of the shapes in
the fourth row?



Answer

What are the observed similarities and differences?

- **Similarity:**
  The unique shapes and colors and direction of the shift (right shift in shapes of every row)
- **Difference:**
  The position of shapes in each row



Pattern recognition in **problem solving is key to determine appropriate solutions** to problems and knowing how to solve certain types of problems. **Recognizing a pattern, or similar characteristics helps break down the problem and also build a construct as a path for the solution**. Ever find yourself saying, 'where have I seen this before', could be a significant step in computational thinking.

Case Study

Case Study 1

Objective:

Create a program to convert temperature within given scale.

```
Program untuk konversi suhu
==============================
1. Celcius
2. Reamur
3. Kelvin
4. Fahrenheit
Derajat suhu asal: 100
Satuan suhu asal (1/2/3/4): 1
Satuan suhu tujuan (1/2/3/4): 4
Hasil konversi dari 100.0 C adalah 212.0 F
```

```
Program untuk konversi suhu
==============================
1. Celcius
2. Reamur
3. Kelvin
4. Fahrenheit
Derajat suhu asal: 100
Satuan suhu asal (1/2/3/4): 2
Satuan suhu tujuan (1/2/3/4): 5
Satuan tidak dikenali
```

Answer

Objective:

Create a program to convert temperature within given scale.

Algorithm:

1. Display the menu
2. Input the temperature
3. Input the source scale
4. Input the destination scale
5. Check the scale's validity
6. If scales are valid, convert the temperature accordingly (this step can be elaborated) and display the result
7. Else display "Scales are invalid"

```
Program untuk konversi suhu
==============================
1. Celcius
2. Reamur
3. Kelvin
4. Fahrenheit
Derajat suhu asal: 100
Satuan suhu asal (1/2/3/4): 1
Satuan suhu tujuan (1/2/3/4): 4
Hasil konversi dari 100.0 C adalah 212.0 F
```

```
Program untuk konversi suhu
==============================
1. Celcius
2. Reamur
3. Kelvin
4. Fahrenheit
Derajat suhu asal: 100
Satuan suhu asal (1/2/3/4): 2
Satuan suhu tujuan (1/2/3/4): 5
Satuan tidak dikenali
```

Answer

Objective:

Create a program to convert temperature within given scale.

Pseudocode:

1. PRINT the menu
2. INPUT *temperature*
3. INPUT *source_scale*
4. INPUT *destination_scale*
5. IF 1 <= *source_scale* <= 4 and 1 <= *destination_scale* <= 4, THEN
6. CONVERT *temperature* accordingly (this step can be elaborated)
7. DISPLAY *result*
8. ELSE DISPLAY “Scales are invalid”

```
Program untuk konversi suhu
==============================
1. Celcius
2. Reamur
3. Kelvin
4. Fahrenheit
Derajat suhu asal: 100
Satuan suhu asal (1/2/3/4): 1
Satuan suhu tujuan (1/2/3/4): 4
Hasil konversi dari 100.0 C adalah 212.0 F
```

```
Program untuk konversi suhu
==============================
1. Celcius
2. Reamur
3. Kelvin
4. Fahrenheit
Derajat suhu asal: 100
Satuan suhu asal (1/2/3/4): 2
Satuan suhu tujuan (1/2/3/4): 5
Satuan tidak dikenali
```

```c
#include <stdio.h>

int main() {
    int asal, tujuan;
    float suhu, hasil;

    printf("Program untuk konversi suhu\n");
    printf("==========================\n");
    printf("1. Celcius\n");
    printf("2. Reamur\n");
    printf("3. Kelvin\n");
    printf("4. Fahrenheit\n");

    printf("Derajat suhu asal: ");
    scanf("%f", &suhu);

    printf("Satuan suhu asal (1/2/3/4): ");
    scanf("%d", &asal);

    printf("Satuan suhu tujuan (1/2/3/4): ");
    scanf("%d", &tujuan);

    float celcius;
    switch (asal) {
        case 1: celcius = suhu; break;
        case 2: celcius = suhu * 5.0 / 4.0; break;
        case 3: celcius = suhu - 273.15; break;
        case 4: celcius = (suhu - 32) * 5.0 / 9.0; break;
        default:
            printf("Satuan asal tidak valid.\n");
            return 1;
    }

    switch (tujuan) {
        case 1: hasil = celcius; break;
        case 2: hasil = celcius * 4.0 / 5.0; break;
        case 3: hasil = celcius + 273.15; break;
        case 4: hasil = (celcius * 9.0 / 5.0) + 32; break;
        default:
            printf("Satuan tujuan tidak valid.\n");
            return 1;
    }

    char *nama_asal, *nama_tujuan;
    switch (asal) {
        case 1: nama_asal = "C"; break;
        case 2: nama_asal = "R"; break;
        case 3: nama_asal = "K"; break;
        case 4: nama_asal = "F"; break;
    }

    switch (tujuan) {
        case 1: nama_tujuan = "C"; break;
        case 2: nama_tujuan = "R"; break;
        case 3: nama_tujuan = "K"; break;
        case 4: nama_tujuan = "F"; break;
    }

    printf("Hasil konversi dari %.1f %s adalah %.1f %s.\n", suhu, nama_asal, hasil, nama_tujuan);

    return 0;
}
```

```
Program untuk konversi suhu
==========================
1. Celcius
2. Reamur
3. Kelvin
4. Fahrenheit
Derajat suhu asal: 100
Satuan suhu asal (1/2/3/4): 1
Satuan suhu tujuan (1/2/3/4): 3
Hasil konversi dari 100.0 C adalah 373.1 K.
```

Case Study 2

Objective:

Create a program to retrieve data from a list within valid index.

```
Masukkan jumlah data:5
Data ke-1:10
Data ke-2:2
Data ke-3:-1
Data ke-4:Lely
Data ke-5:100

Index awal:0
Index akhir:-1
Program hanya menerima indeks positif
Index awal:4
Index akhir:2
Indeks awal harus lebih kecil dari indeks akhir
Index awal:5
Index akhir:10
Indeks melebihi panjang list
Index awal:2
Index akhir:4
Data indeks ke 2 s.d. 4
['-1', 'Lely', '100']
Ulang lagi (y/t)?t
```

Answer

Objective:

Create a program to retrieve data from a list within valid index.

Algorithm:

1. Input the number of data (N)
2. For each data, input the amount of data
3. Input start and end index
4. Check the index validity (this step can be elaborated)
5. If the index is not valid, display the error message and return to step 3
6. Else, display the data according to the given index
7. Check if the user would like to repeat
8. If yes, return to step 3

```
Masukkan jumlah data:5
Data ke-1:10
Data ke-2:2
Data ke-3:-1
Data ke-4:Lely
Data ke-5:100

Index awal:0
Index akhir:-1
Program hanya menerima indeks positif
Index awal:4
Index akhir:2
Indeks awal harus lebih kecil dari indeks akhir
Index awal:5
Index akhir:10
Indeks melebihi panjang list
Index awal:2
Index akhir:4
Data indeks ke 2 s.d. 4
['-1', 'Lely', '100']
Ulang lagi (y/t)?t
```

Answer

Objective:

Create a program to retrieve data from a list within valid index.

Pseudocode:

1. INPUT *N*
2. FOR *index* in 1 to *N*, input *data[index]*
3. INPUT *start_index* and *end_index*
4. IF *start_index* is not VALID or *end_index* is not VALID (this step can be elaborated), THEN
5. DISPLAY the error message
6. RETURN to step 3
7. ELSE DISPLAY *data[start_index:end_index]*
8. IF the user would like to repeat, THEN
9. RETURN to step 3

```
Masukkan jumlah data:5
Data ke-1:10
Data ke-2:2
Data ke-3:-1
Data ke-4:Lely
Data ke-5:100

Index awal:0
Index akhir:-1
Program hanya menerima indeks positif
Index awal:4
Index akhir:2
Indeks awal harus lebih kecil dari indeks akhir
Index awal:5
Index akhir:10
Indeks melebihi panjang list
Index awal:2
Index akhir:4
Data indeks ke 2 s.d. 4
['-1', 'Lely', '100']
Ulang lagi (y/t)?t
```

```c
#include <stdio.h>
#include <string.h>

int main() {
    int jumlah, i, awal, akhir;
    char data[100][100];
    char ulang;

    do {
        printf("Masukkan jumlah data: ");
        scanf("%d", &jumlah);
        getchar();

        for (i = 0; i < jumlah; i++) {
            printf("Data ke-%d: ", i + 1);
            fgets(data[i], sizeof(data[i]), stdin);
            data[i][strcspn(data[i], "\n")] = '\0';
        }

    while (1) {
        printf("\nIndex awal: ");
        scanf("%d", &awal);
        printf("Index akhir: ");
        scanf("%d", &akhir);

        if (awal < 0 || akhir < 0) {
            printf("Program hanya menerima indeks positif\n");
        } else if (awal >= akhir) {
            printf("Index awal harus lebih kecil dari indeks akhir\n");
        } else if (awal >= jumlah || akhir > jumlah) {
            printf("Indeks melebihi panjang list\n");
        } else {
            printf("Data indeks ke %d s.d. %d\n", awal, akhir);
            for (i = awal; i <= akhir; i++) {
                printf("'%s'%s", data[i], (i < akhir) ? ", " : "\n");
            }
            break;
        }
    }

        printf("Ulang lagi (y/t)? ");
        getchar();
        scanf("%c", &ulang);
        getchar();

    } while (ulang == 'y' || ulang == 'Y');

    return 0;
}
```

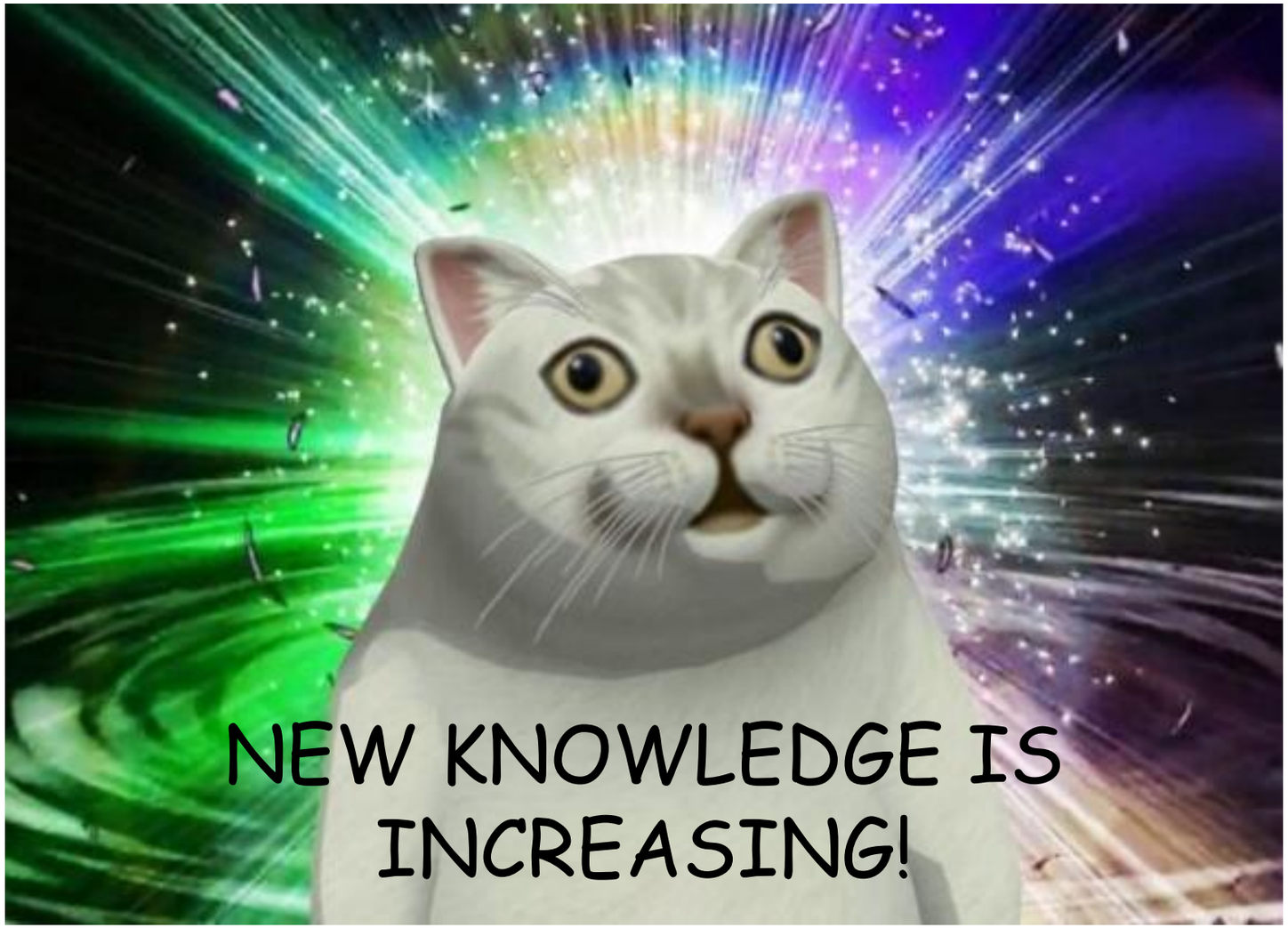
```
Masukkan jumlah data: 10
Data ke-1: 2
Data ke-2: 4
Data ke-3: 8
Data ke-4: yunia
Data ke-5: 90
Data ke-6: 76
Data ke-7: 4
Data ke-8: ==
Data ke-9: jhuh
Data ke-10: hhjhj

Index awal: 0
Index akhir: 9
Data indeks ke 0 s.d. 9
'2', '4', '8', 'yunia', '90', '76', '4', '==', 'jhuh', 'hhjhj'
Ulang lagi (y/t)? t
```

TASK

Buatlah flowchart, algoritma dan pseudocode dari output berikut

```
Program Konversi Sudut Derajat ke Radian
========================================
Masukkan nilai sudut dalam derajat: 80
Hasil konversi: 1.396263 radian
```


NEW KNOWLEDGE IS
INCREASING!

Reference

- Karl, Beecher. "Computational Thinking: A Beginner's Guide to Problem-Solving and Programming." Swindon, UK: BCS, The Chartered Institute for IT (2017).

THANK YOU!