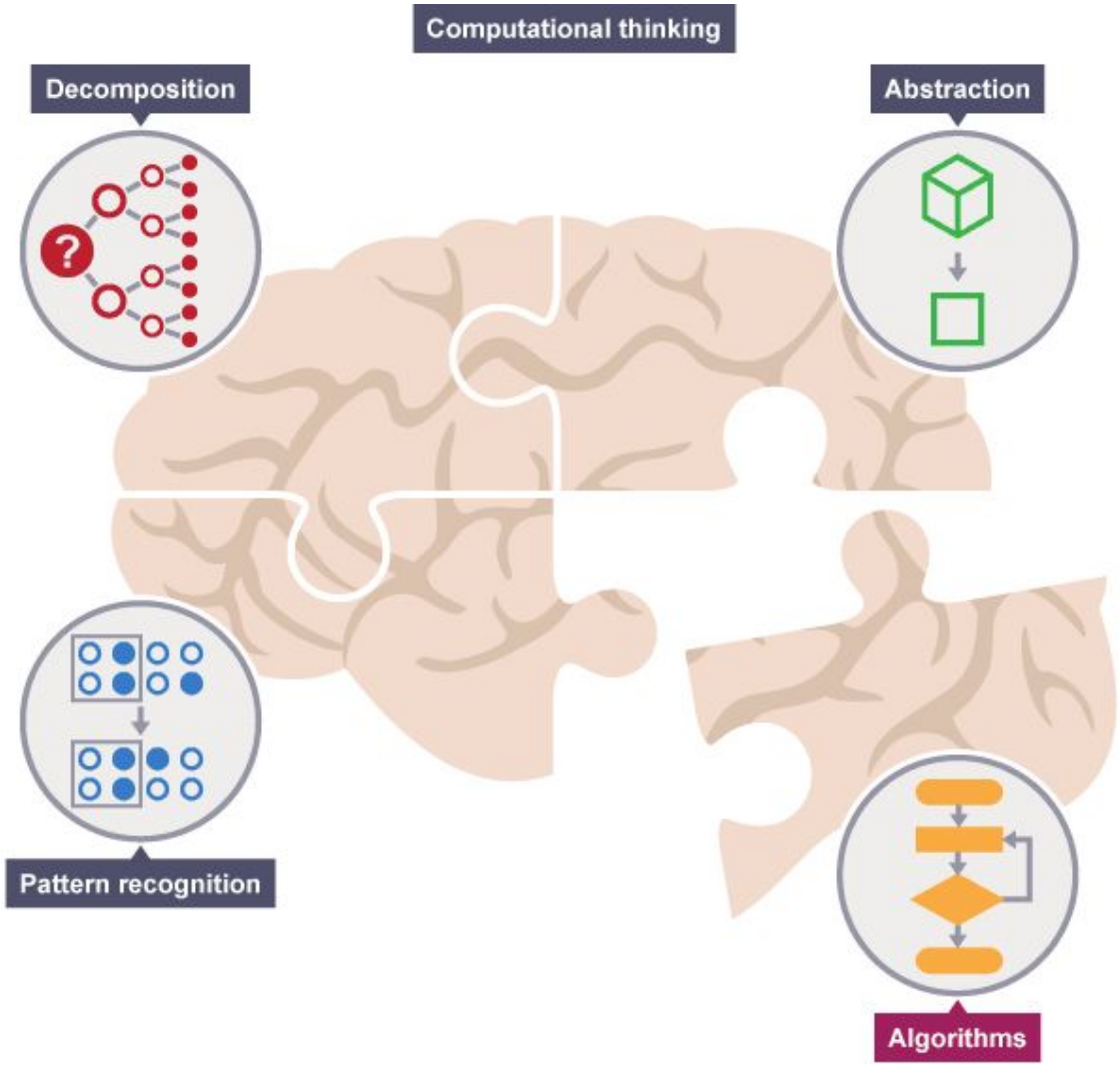
CT #4 Algorithms

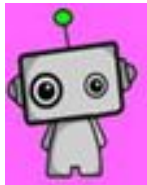
Logika Pemrograman (SIF104)

Program Studi Informatika - Program Sarjana
Fakultas Teknologi Industri



Algoritma adalah sekumpulan instruksi langkah demi langkah untuk memecahkan masalah

Application of Computational Thinking Across the Curriculum!



Algorithmic Design

Create a set of step-by-step instructions to complete a task

Primary Terminology – "Instructions"

Early Years Foundation Terminology – "Plan"

Draw a map

Explain the process of photosynthesis

Write a piece of music

Write an algorithm to show how your computer game character will move.

Write a short term training programme

Create a timeline of events for WWII

Create a paint by numbers

Making patterns

Create a storyboard for an animation

Create a phrase book

Write a recipe for ...

Write out the steps for conducting your experiment

Choreograph a dance / gymnastics routine

Create a tactical playbook

Create an origami

Dot to dot

Write a shopping list

Build a pirate ship out of Lego

Create a blueprint to design a ...

Draw a diagram to show the water cycle

Create a coaching card for the tennis serve

Create a family tree

Create a flowchart to show how you would ...

Speed cup stacking!

Create a how 2 guide so someone else can recreate your drawing

Apa itu Algoritma?

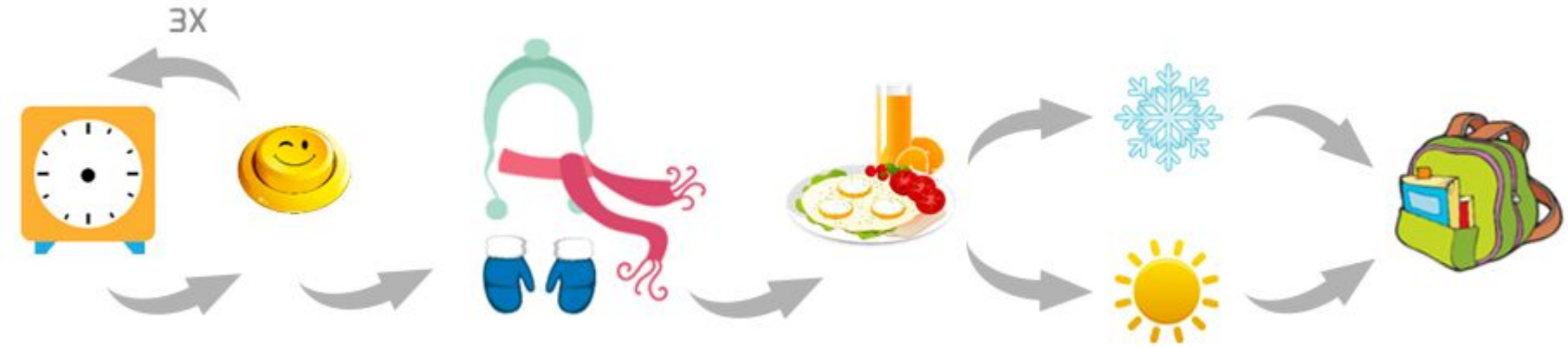
- Tersusun dari instruksi-instruksi yang sudah teridentifikasi tujuannya dan mempunyai urutan pelaksanaannya.
- Algoritma sering digunakan sebagai ***starting point*** untuk membuat program komputer, terkadang ditulis dalam suatu ***flowchart*** atau ***pseudocode***.
- Komputer hanya ***sebagus*** algoritmanya.
  Kalau algoritma buruk, hasilnya juga buruk.
- Algoritma digunakan untuk berbagai hal:
  perhitungan, ***pemrosesan data***, ***automasi***, dll

Definisi Algoritma

- Teknik penyusunan ***langkah-langkah penyelesaian masalah*** dalam ***bentuk kalimat*** dengan ***jumlah terbatas*** tetapi tersusun secara ***logis*** dan ***sistematis***.
- Suatu prosedur yang ***jelas*** untuk ***menyelesaikan suatu persoalan*** dengan menggunakan ***langkah-langkah*** tertentu dan ***terbatas*** jumlahnya.
- Susunan langkah yang pasti, yang bila diikuti maka akan mentransformasikan data ***input*** menjadi ***output*** yang berupa informasi.

Algoritma

- Algoritma adalah urutan ***langkah-langkah logis*** penyelesaian masalah yang disusun secara ***sistematis*** dan ***logis***.
- Kata ***logis*** merupakan ***kata kunci*** dalam algoritma.
- Urutan langkah logis, yang berarti ***algoritma harus mengikuti suatu urutan tertentu, tidak boleh melompat-lompat***.



Kriteria Algoritma

- Langkah-langkah dalam algoritma harus dapat ditentukan bernilai ***benar*** atau ***salah***
- Kriteria Algoritma:
  - ***Input***: dapat memiliki nol atau lebih masukan
  - ***Output***: harus memiliki ***minimal*** satu buah keluaran yang dihasilkan
  - ***Definiteness*** (pasti): memiliki harus instruksi-instruksi yang jelas dan tidak ***ambigu***.
  - ***Finiteness*** (ada batas): harus memiliki titik berhenti (***stopping role***).
  - ***Effectiveness*** (tepat dan efisien): sebisa mungkin harus dapat dilaksanakan dan efektif

Contoh Algoritma

- Menentukan apakah suatu bilangan merupakan bilangan ***ganjil*** atau bilangan ***genap***.
  - ***Masukkan*** sebuah bilangan sembarang
  - ***Bagi*** bilangan tersebut dengan bilangan 2
  - ***Hitung*** sisa hasil bagi pada langkah 2
  - Bila ***sisa*** hasil bagi sama dengan 0 maka bilangan itu adalah bilangan *genap* tetapi bila sisa hasil bagi sama dengan 1 maka bilangan itu adalah bilangan *ganjil*

Representasi Algoritma: Pseudocode

- Bahasa pemrograman memiliki ***sintaks*** khusus yang harus digunakan agar program dapat berjalan dengan baik.
- ***Pseudocode*** **bukanlah** bahasa pemrograman.
- Pseudocode adalah cara sederhana untuk ***mendeskripsikan*** serangkaian ***instruksi*** tanpa perlu sintaks tertentu.
- Di dalam pseudocode memuat ***logika*** penyelesaian masalah
- Pseudocode harus berisi bahasa yang singkat, padat dan jelas

Pseudocode

- Contoh pseudocode sederhana:

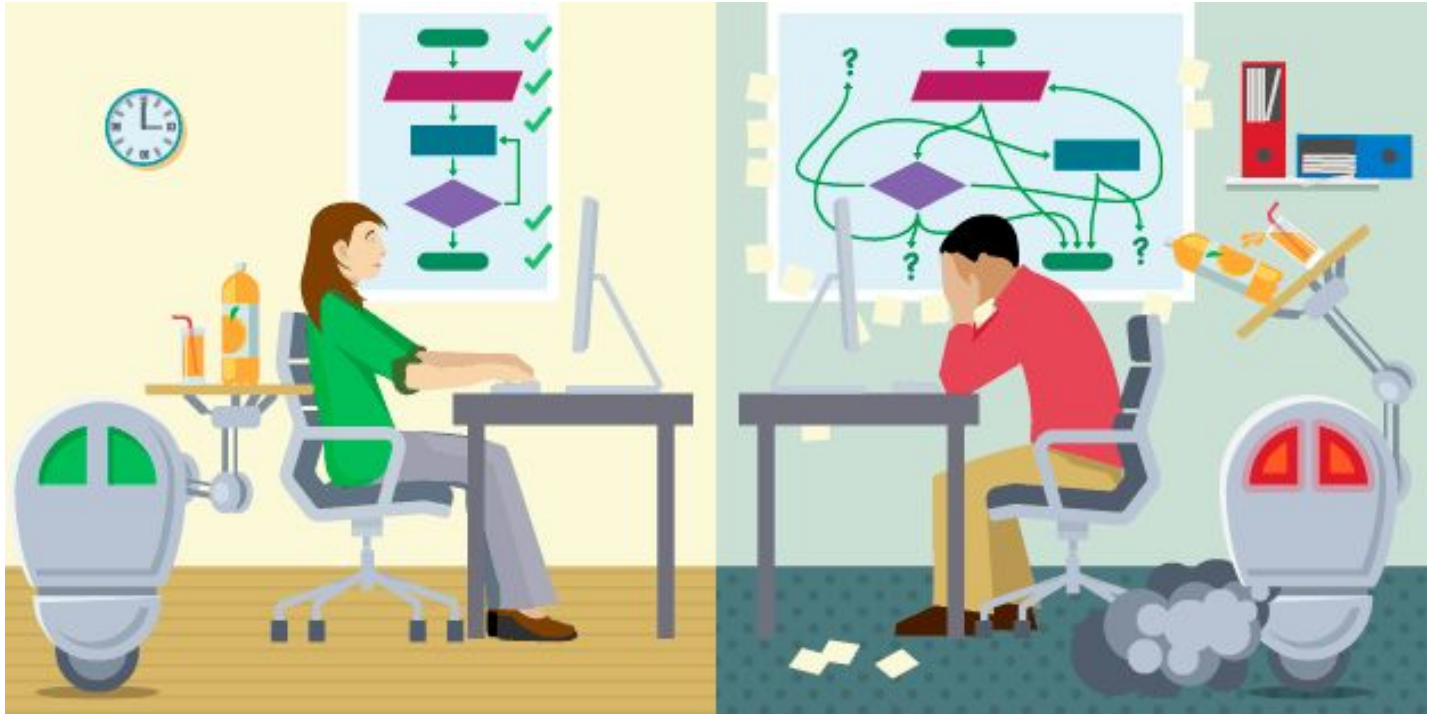
```
Mulai
Input a
Input b
c = a + b
Print "c"
Selesai
```

Pseudocode

- Contoh pseudocode sederhana:

```
OUTPUT ”Siapa namamu?”
INPUT pengguna memasukkan namanya
STORE masukan pengguna disimpan dalam variabel nama
OUTPUT ”Haloo ” + nama
```

Mengevaluasi Solusi



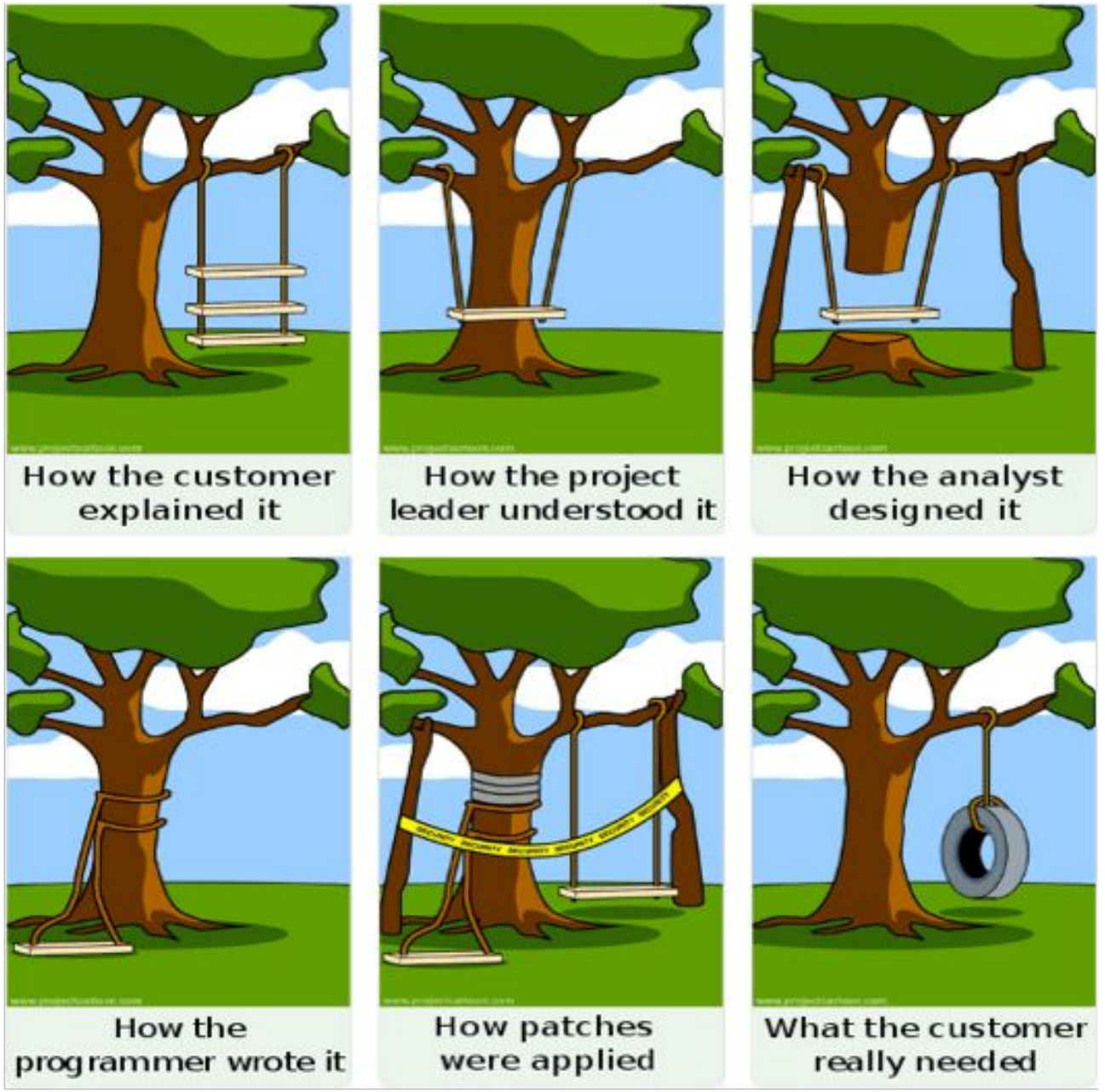
Kegagalan dalam *evaluasi* bisa mempersulit dalam penulisan kode program.

Setelah algoritma ditulis, perlu dilakukan pengecekan:

- Apakah mudah dipahami? Apakah telah didekomposisi dengan baik?
- Apakah sudah lengkap? Sudah menyelesaikan semua aspek masalah?
- Apakah efisien? Secepat mungkin atau hemat sumber daya?
- Apakah telah memenuhi kriteria desain awalnya?

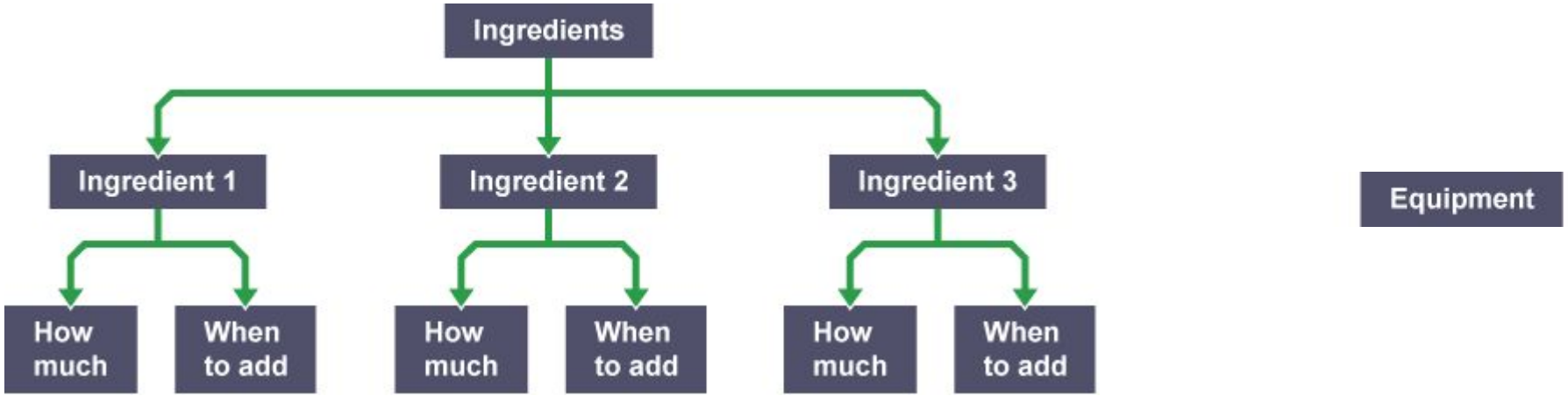
Kenapa perlu mengevaluasi solusi?

- Jika ada ***kesalahan*** dalam solusi, akan sulit untuk membuat programnya.
- Kemungkinan ***terburuk***, program tidak akan menyelesaikan masalah dengan benar.
- Kemungkinan penyebab kesalahan:
  - masalah tidak dipahami seutuhnya;
  - dekomposisi masalah kurang tepat
  - solusi tidak lengkap;
  - beberapa bagian dari masalah belum diselesaikan
  - tidak efisien;
  - solusi terlalu kompleks atau terlalu panjang
  - solusi tidak sesuai dengan kriteria desain awal;
  - tidak cocok untuk tujuan

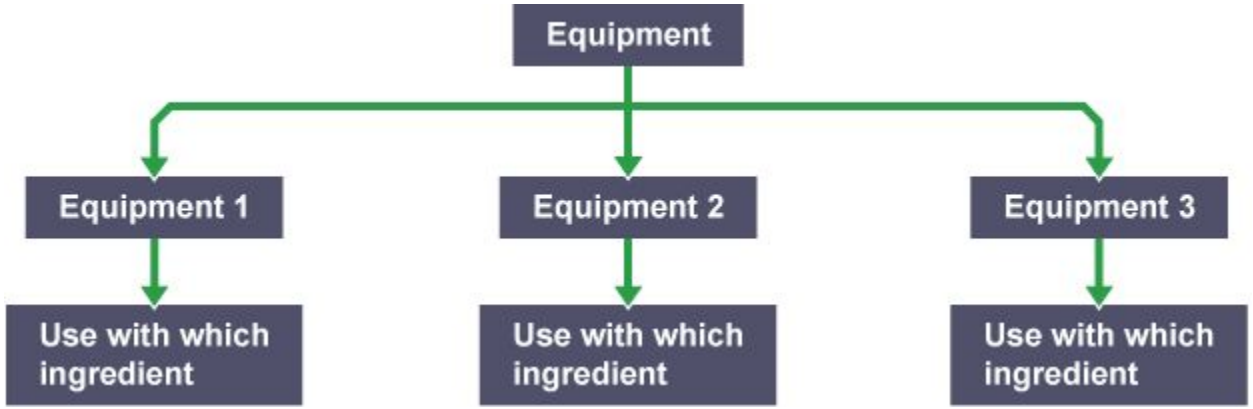

How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
How patches were applied
What the customer really needed

Masalah yang tidak terdekomposisi dengan baik

(Studi Kasus: Membuat Kue)



Idealnya, "***Equipment***" perlu diuraikan lebih lanjut untuk menyatakan peralatan mana yang diperlukan untuk tiap bahan yang digunakan.



Solusi yang tidak lengkap

(Studi Kasus: Membuat Kue)

Hasil dekomposisi masalah:

- Mau buat kue apa?
- Bahan apa yang diperlukan?
- Kuenya untuk berapa orang?
- Berapa lama memanggang kuenya?
- Peralatan apa saja yang dibutuhkan?

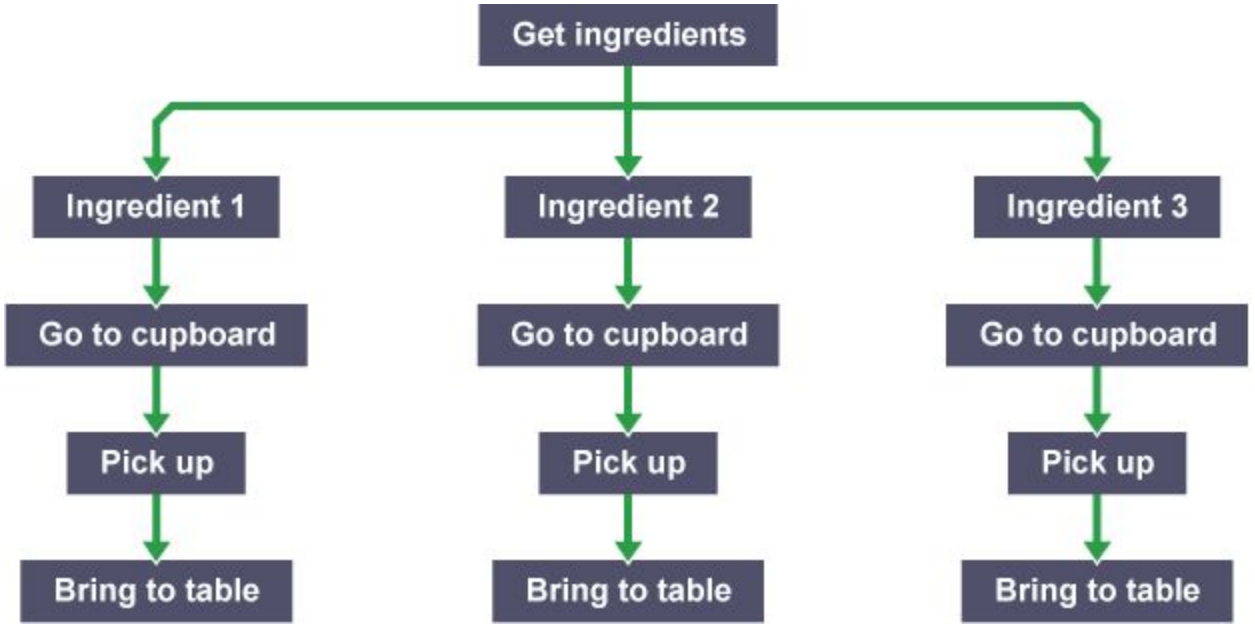
Masih belum lengkap!

Kita perlu tahu:

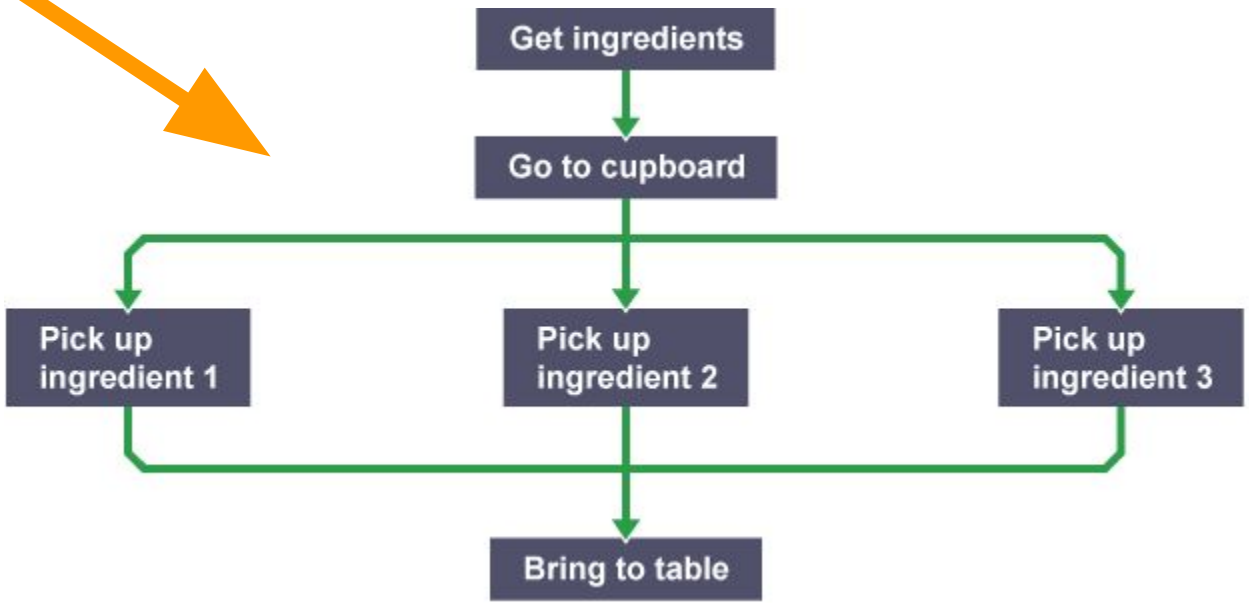
- Memanggang kue di mana?
- Berapa suhu ketika memanggang kuenya?

Solusi yang tidak efisien

(Studi Kasus: Membuat Kue)



dari 9 langkah,
direduksi jadi 5 langkah saja



Solusi yang tidak memenuhi kriteria desain awal

(Studi Kasus: Membuat Kue)

Contoh:

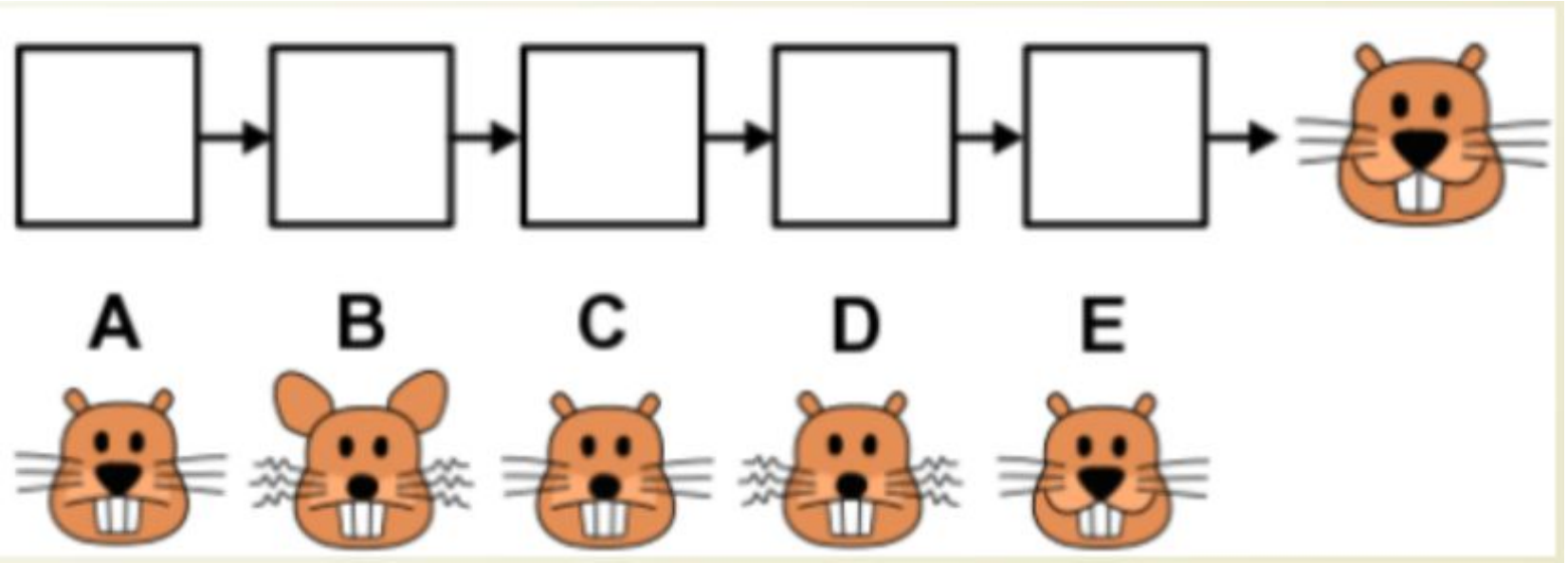
Desain awal: Kita mau buat kue tart **cokelat** dengan ***icing*** cokelat dan taburan chocochips dan stroberi (salah satu)

Solusi: menaburkan parutan keju di atas kue

Hasil: seperti gambar

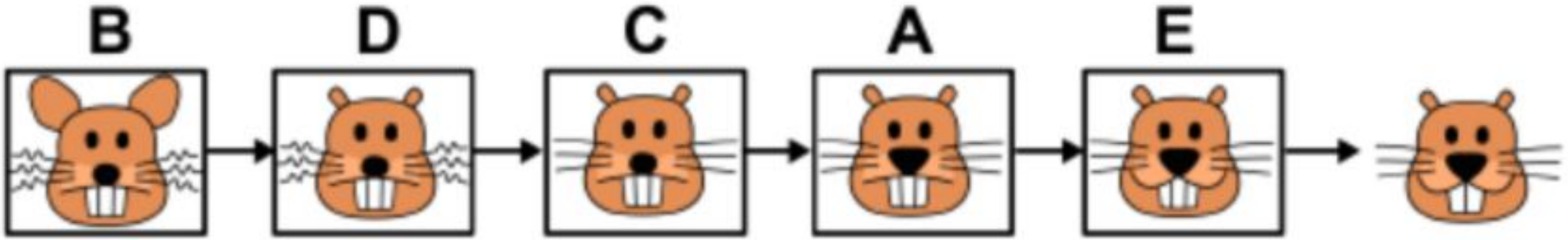
UNIVERSITAS ISLAM INDONESIA

Contoh Implementasi CT



Taro berencana membuat animasi wajah yang terdiri dari urutan gambar-gambar. Agar animasi berjalan mulus, hanya satu fitur pada wajah yang berubah dari suatu gambar ke gambar sebelahnya. Sayangnya, gambar-gambar yang ada bercampur baur. Taro harus menemukan urutan gambar yang benar. Untungnya, Taro tahu gambar terakhir yang mana. Bantu urutkan ya!

Jawaban

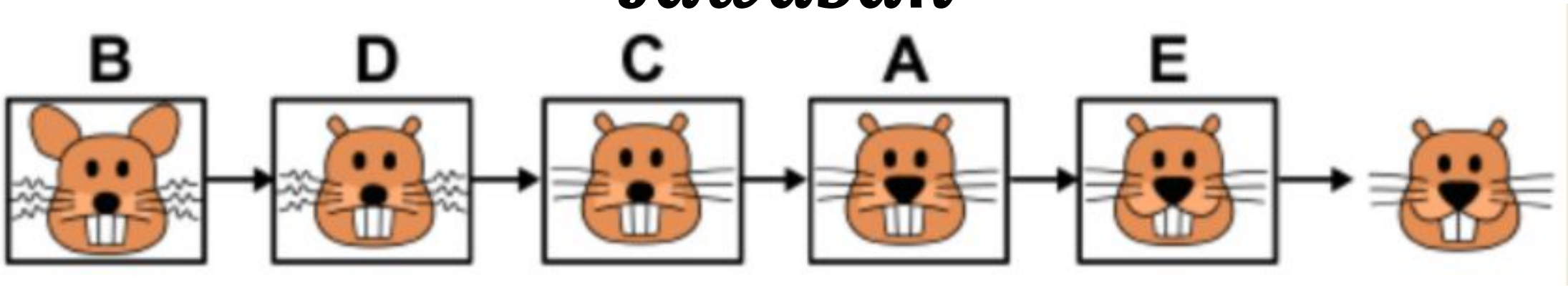


Skills: Decomposition, Pattern Recognition, Abstraction, Algorithm

Wajah Taro harus dikenali bagian-bagian penyusunnya dengan dekomposisi & pattern recognition.

Namun harus diingat, jika untuk menemukan perbedaan di antara gambar-gambar tersebut, harus ditemukan atribut penting wajah dengan abstraksi.

Jawaban



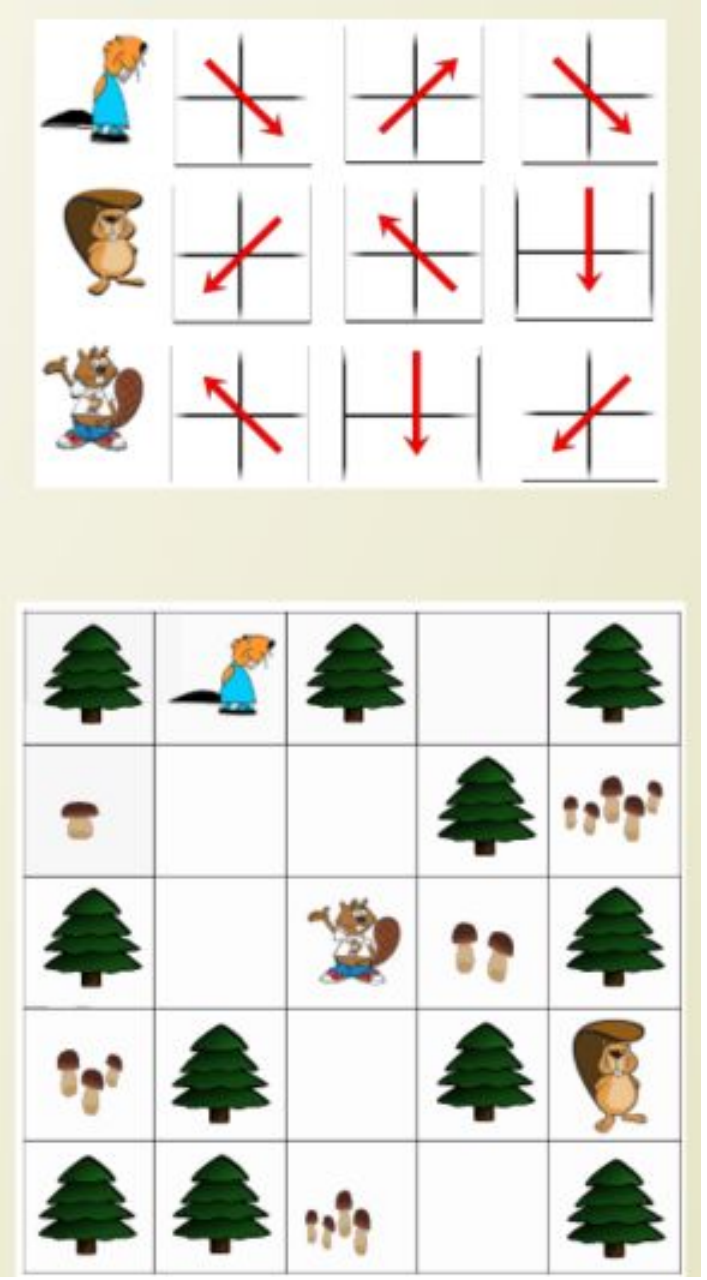
Skills: Decomposition, Pattern Recognition, Abstraction, Algorithm

Daftar atribut & kemungkinan nilainya adalah

- telinga (kecil & besar)
- mulut (datar & senyum)
- hidung (kecil & besar)
- banyak gigi (2 & 3)
- kumis (lurus & keriting)

Langkah terakhir adalah menyusun urutan dengan benar.

Jamur

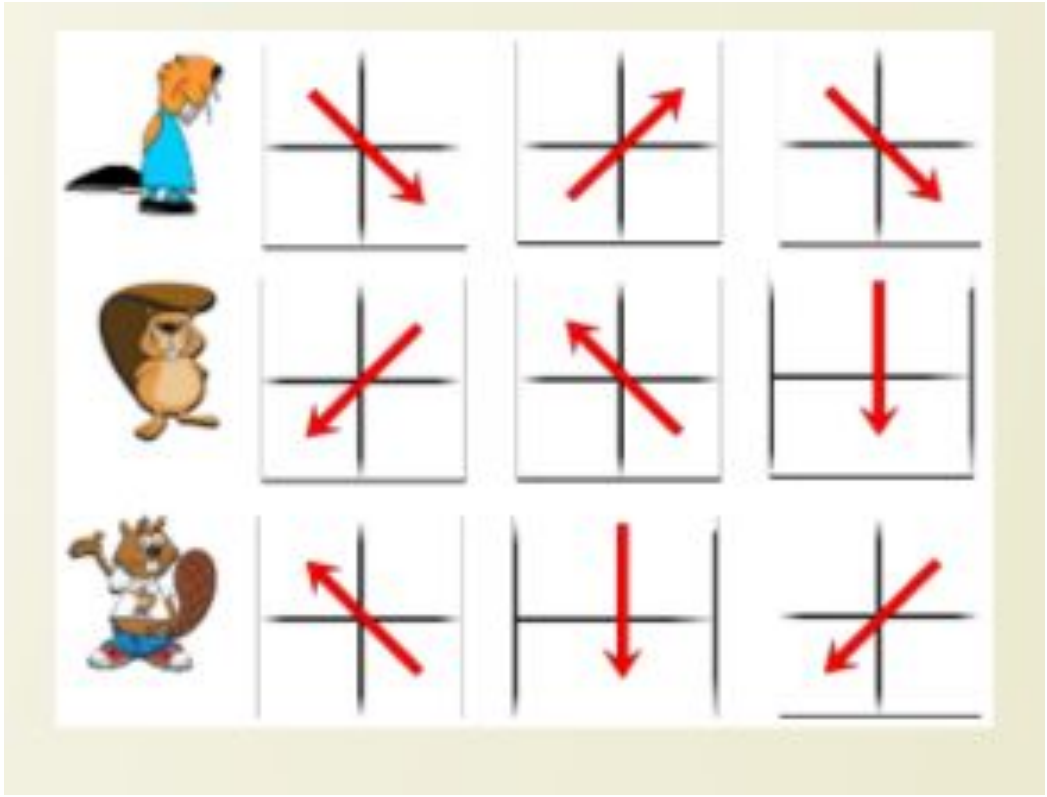


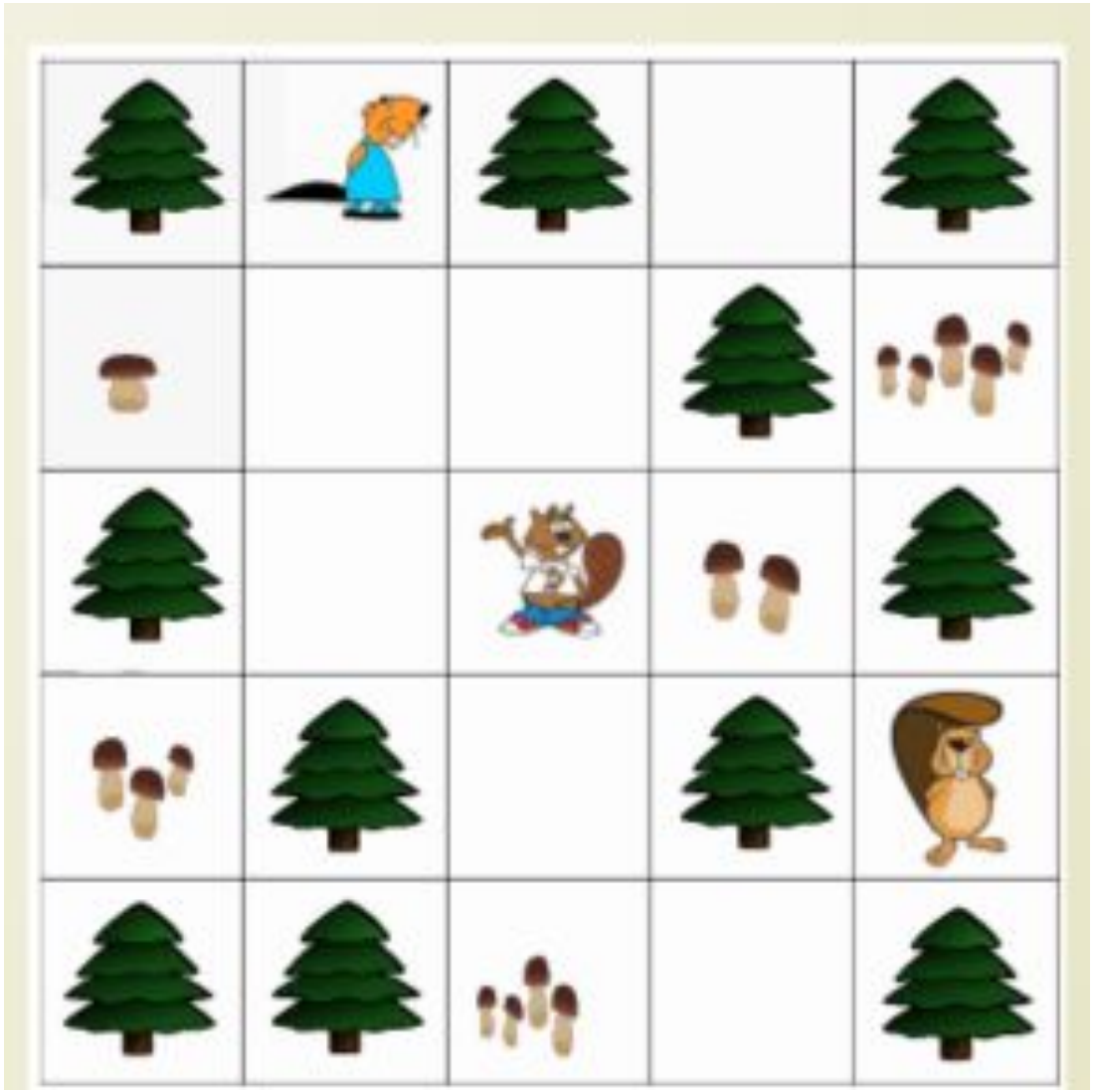
Tiga berang-berang berdiri di dalam hutan. Tiap berang-berang ingin pergi ke tempat di mana ada jamur. Panah di gambar kanan atas menunjukkan arah berang-berang akan berjalan.

Di mana lokasi akhir berang-berang?

Skill: Algorithmic Thinking

Jamur



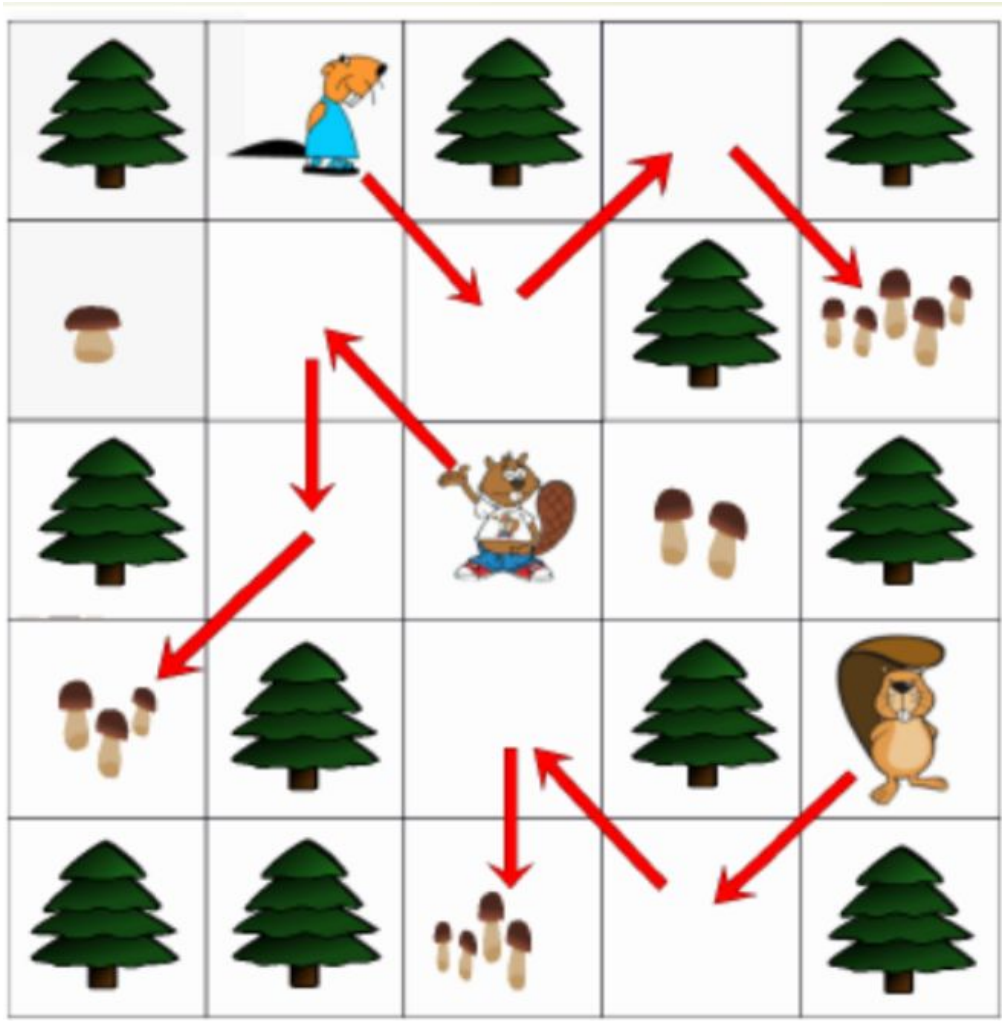


Jamur

Jawaban ada di gambar sebelah kanan.

Skill: Algorithmic Thinking

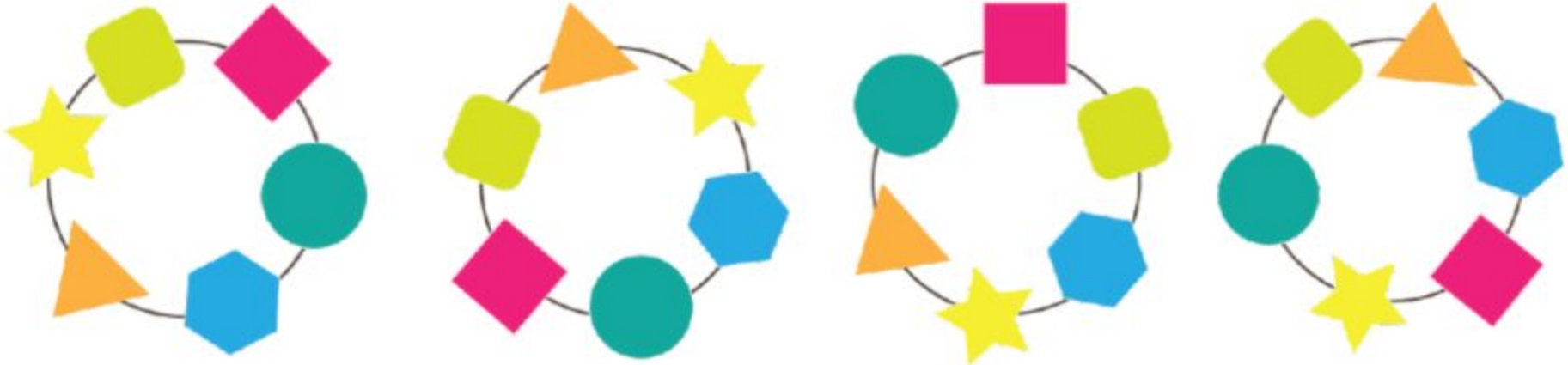
Kumpulan instruksi sederhana yang disebut ***algoritma*** yang dapat membantu kita memecahkan masalah. *Terkadang lebih mudah melakukan ini dengan gambar dan panah daripada dengan kata-kata.*



Emily has broken her favourite bracelet. The broken bracelet now looks like this:

Question:

Which of the following four bracelets shows what the bracelet looked like when it was whole?



Untuk menyelesaikan masalah ini, kita perlu meng***evaluasi*** keempat pilihan untuk menentukan mana gelang yang tepat.

In the Stack Computer, calculations are entered in a different way to a normal calculator.

Examples:

2+3 must be entered as 2 3 +

10-2 must be entered as 10 2 -

5*2+3 must be entered as 5 2 * 3 +

5+2*3 must be entered as 5 2 3 * +

(8-2)*(3+4) must be entered as 8 2 - 3 4 + *

In this task **decomposition** is required to break up the task and to deal with one section at a time.

Question:

How should the following computation be entered: 4*(8+3)-2?