* What is an "algorithm"?

some descriptions to get something done (correctly/efficiently) by a computer.

algorithm ⟺ 程式譜
食譜
樂譜
陣譜
劍譜
暗器譜

* five criteria of 食譜

◎蕃茄炒蛋（阿基師）

㊍蕃茄*3, 蛋*3, 蔥*2, 薑一小塊, 太白粉, 鹽, 糖

① 切蔥花, 薑末備用
② 蕃茄去蒂畫十字刀, 汆燙, 下冷水去皮後切開
③ 打蛋加少鹽
④ 起油鍋, 爆香薑末, 加入蕃茄, 水, 鹽, 糖
⑤ 加太白粉勾芡
⑥ 加蛋翻炒
⑦ 撒上蔥花

Input : ㊍
Output : ◎ (correctness)
Definiteness : clear ① ~ ⑦ (do-able by cook)
Finiteness : can be finished (will not cook forever)
Effectiveness : do-able by "cookwares"

* five criteria of algorithm (Knuth)
  程式讀
  Input
  Output
  Definiteness : do-able by programmars
  Finiteness : can end
  Effectiveness : do-able by computers

* int getMinPos(int* arr, int len) {
  ☆
  int minpos = 0; int i;
  for(i=1; i<len; i++){
  if(arr[i] < arr[minpos])
  minpos = i;
  }
  return minpos;
  ◎
  △
  }

Input : ☆
Output : ◎
Definiteness : △ can be "coded"
Finiteness : finiteness of every step in △
Effectiveness : △ can be compiled and executed.

* Correctness of getMinPos

Claim: the function returns m such that

$arr[m] \leq arr[j]$ for $j = 1, 2, \cdots, len-1$

⇑

claim 2: at the end of loop with $i = k$,

$arr[minpos] \leq arr[j]$ for $j = 0, 1, 2, \cdots, k$

⇑ Math Induction

at the end of $i = 1$

if minpos is 0

$\Leftrightarrow arr[1] \geq arr[0] = arr[minpos]$

if minpos is 1

$\Leftrightarrow arr[1] < arr[0]$

$=$

$arr[minpos]$

so claim 2 is true

at $k = 1$

"assume" claim 2 is true

at $k = \square - 1$

when $k = \square$

if minpos is not $\square$

$\Leftrightarrow arr[\square] \geq arr[minpos]$

$arr[j] \geq arr[minpos]$

for $j = 0, 1, 2, \cdots, \square - 1$

if minpos is $\square$

$\Leftrightarrow arr[\square] < arr[old\ minpos]$

$arr[minpos]$

$\leq arr[j]$

for $j = \cdots$

so claim 2 is true

at $k = \square - 1$

claim 2 often called as

"invariance" property of loop