บทที่ 3
อัลกอริทึม (Algorithm)

เนื่องจากอัลกอริทึมเป็นพื้นฐานการนำไปสู่โปรแกรมที่ดี ดังนั้น การศึกษาประสิทธิภาพและความถูกต้องของอัลกอริทึมจึงเป็นสิ่งที่จำเป็นและสำคัญเพื่อที่จะนำไปสู่การพัฒนาคุณภาพของโปรแกรม ตลอดจนการเลือกภาษาคอมพิวเตอร์เพื่อดำเนินการต่อไป แต่อัลกอริทึมก็ไม่ใช่เป็นเพียงเงื่อนไขเดียวเท่านั้น ที่จะทำให้โปรแกรมที่เขียนขึ้นมานั้นเป็นโปรแกรมที่ดีจริงๆแล้วยังจะต้องมีปัจจัยอย่างอื่นประกอบอยู่ด้วยดังคำกล่าวที่ว่า

" A good algorithm is a necessary condition but not a sufficient condition for a good program "

อัลกอริทึมคือขั้นตอนการดำเนินงานซึ่งเราเลือกขึ้นมาเพื่อใช้ตอนที่เราจะเขียนโปรแกรมจะเห็นได้ว่าขั้นตอนการทำงานของคอมพิวเตอร์อันนำไปสู่เป้าหมายที่ต้องการมีอยู่หลายรูปแบบ แต่ละรูปแบบอาจจะแตกต่างกันแต่ก็สามารถดำเนินงานให้บรรลุวัตถุประสงค์เดียวกันได้ ดังนั้นเราจึงจะต้องเลือกเอาอัลกอริทึมที่ดีและมีประสิทธิภาพที่สุดไปใช้งานในกรณีย์ที่ผู้ใช้สามารถจะแจงปัญหาที่ต้องการดำเนินการให้ออกมาเป็นรูปของอัลกอริทึมที่ชัดเจนได้ก็จะเป็นการดีที่จะช่วยให้การออกแบบโปรแกรมเป็นไปได้อย่างสะดวกและง่ายขึ้น ตัวอย่างต่อไปนี้จะแสดงวิธีการเลือกอัลกอริทึมอย่างง่ายๆ

เช่น ถ้าเราจะคำนวณค่าของ y จากฟังก์ชันต่อไปนี้คือ

$$Y = AX^3 + BX^2 + CX + D$$

ในกรณีย์ที่เราใช้ภาษาฟอร์แทรนในการดำเนินงาน เราอาจจะเขียนคำสั่งนี้ออกมาเป็นรูปแบบดังนี้

```
Y = A * X ** 3 + B * X ** 2 + C * X + D
```

หรือในกรณีย์ณีที่บางภาษาไม่มีเครื่องหมายยกกำลังให้ใช้ ก็อาจจะใช้วิธีการง่ายๆ คือ

```
Y = A * X * X * X + B * X * X + C * X + D
```

ซึ่งในรูปแบบนี้จะเห็นได้ว่าในนิพจน์คณิตศาสตร์มีการหาค่าของ y ได้จะเกิดจากการ คูณกัน 6 ครั้ง และการบวก 3 ครั้งดังนั้นเราอาจจะอาศัยหลักการทางคณิตศาสตร์มาช่วยปรับตัวแบบของนิพจน์คณิตศาสตร์นี้ให้ดีขึ้นดังนี้

$$Y = AX^3 + BX^2 + CX + D$$
$$= X(AX^2 + BX + C) + D$$
$$= X(X(AX + B) + C) + D$$

ดังนั้นเราจะได้สมการใหม่ในการหาค่าของ y ได้ดังนี้

$$Y = X * (X * (A * X + B) + C) + D$$

ซึ่งจะเห็นได้ว่ารูปแบบนี้จะดำเนินการได้โดยใช้การคูณเพียง 3 ครั้ง และการบวก 3 ครั้ง นอกจากการ ลด จำนวนครั้งของการปฏิบัติทางคณิตศาสตร์นี้แล้วอัลกอริทึมยังช่วยเพิ่มประสิทธิภาพ ของความถูกต้องด้วยทั้งนี้เพราะการคูณจำนวนที่เป็น real แต่ละครั้งจะมีการปัดเศษอยู่ด้วยเสมออันเนื่องมาจากข้อจำกัดในการเก็บข้อมูลในสมองของคอมพิวเตอร์ดังนั้นการที่เราลดจำนวนครั้งของปฏิบัติการทางคณิตศาสตร์จึงเท่ากับเพิ่มความถูกต้องของการประเมินผลนิพจน์ทางคณิตศาสตร์ให้มากขึ้นด้วย

จากตัวอย่างนี้จะนำไปสู่การเลือกรูปแบบอัลกอริทึมแบบอื่นๆในทางคณิตศาสตร์ลักษณะอื่นๆเช่น การหาค่า Prime Number ถ้าเราต้องการพิสูจน์ว่า N เป็น Prime Number หรือไม่นั้น หลักการในทางคณิตศาสตร์มีอยู่ว่า N จะเป็น Prime Numberได้ก็ต่อเมื่อตัวที่มาหาร N ได้จะมีแต่เลข 1 กับ N เท่านั้น

ตัวอย่างของ Prime Number เช่น 3,5,7,11,13,...เป็นต้น ดังนั้นเพื่อเป็นการพิสูจน์ว่าเลขจำนวน N ใดๆ ที่กำหนดมาให้นั้นจะมีคุณสมบัติเป็น Prime Number ได้เราจะต้องทดลองนำเลขที่เล็กกว่า N มาหารกับ N เลขเซตที่จะหารกับ N นั้นจะประกอบด้วยเลข { 2,3,...,N-1 } ดังนั้นอัลกอริทึมที่จะดำเนินการก็คือทดลองว่าเริ่มตั้งแต่เลขเป็นต้นไปว่ามีตัวใดจะหาร N ได้หรือไม่ถ้าหารได้ลงตัวก็ตอบว่า N ไม่เป็น Prime Number จึงหยุดการหารแต่ถ้าหากหารไม่ได้ลงตัวก็จะขยับค่าตัวหารไปอีก 1 แล้วดำเนินการในลักษณะเดิมไม่เราก็ตามที่ตัวหารนั้นขยับไปจนเป็นค่า N-1 แสดงว่าไม่มีเลขตัวใดที่น้อยกว่า N และหารN ลงตัวก็จะพิสูจน์ได้ว่า N เป็น Prime Number ถ้าเราพิจารณาเซตของเลขที่เป็นตัวหาร { 2,3,...,N-1 } จะพบได้ว่าเซตดังกล่าวมีเลขที่เป็นตัวร่วมอยู่ประกอบกันอยู่ เช่น เลข 2,4,6,8,... หรือตัวประกอบของเลข

3,9,21,24,...นั่นหมายความถ้าเราเอาเลข2ไปหารNไม่ลงตัวก็หมายความว่าเราเอาเลข 4,6,8,...ไปหาร N ย่อมไม่ลงตัวเช่นกัน แต่ถ้าเราเลือกอัลกอริทึม โดยใช้เลข possible value set อันประกอบด้วยเซต { 2,3,4,5,... } ก็จะเป็นวิธีการที่ด้อยประสิทธิภาพดังเหตุผลที่กล่าวมา แล้วว่าแต่วิธีการของอัลกอริทึมดังกล่าวก็ถือว่าเป็น common first algorithm ของการพิสูจน์ว่า N เป็น Prime Number หรือไม่ อัลกอริทึมดังกล่าวเขียนเป็นผังโปรแกรมได้ดังนี้

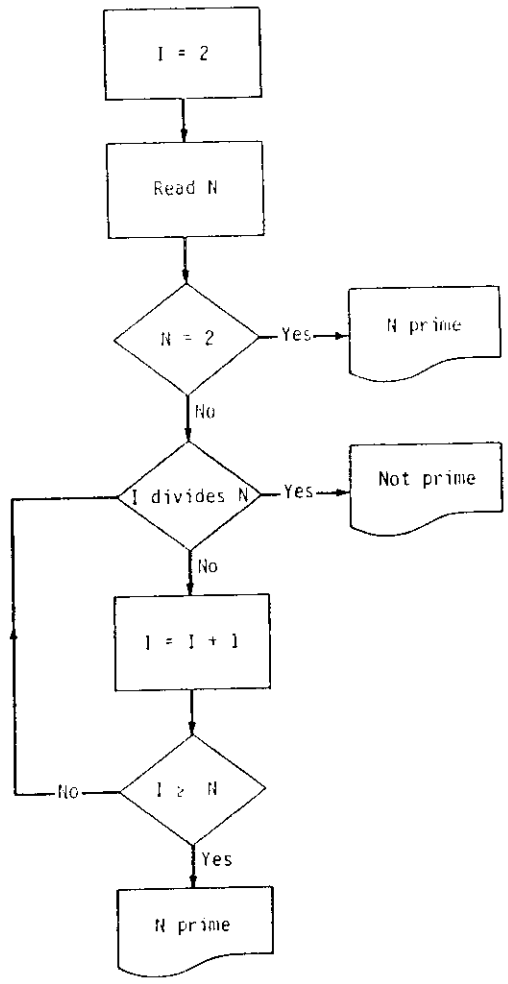

Figure Algorithm 1

จาก common algorithm ที่กล่าวมานี้เราจะพัฒนาไปเลือกอัลกอริทึมใหม่ซึ่งจะนำค่า 2 ไปหาร N และค่าถัดไปที่หาร N จะเป็นเลขคี่ (add number) เท่านั้นด้วยกระบวนการ ดังกล่าวเราจะลดภาระงานไปได้เกือบครึ่งดังนั้นอัลกอริทึมที่เราพัฒนาใหม่จะออกเป็น program flow ดังนี้คือ

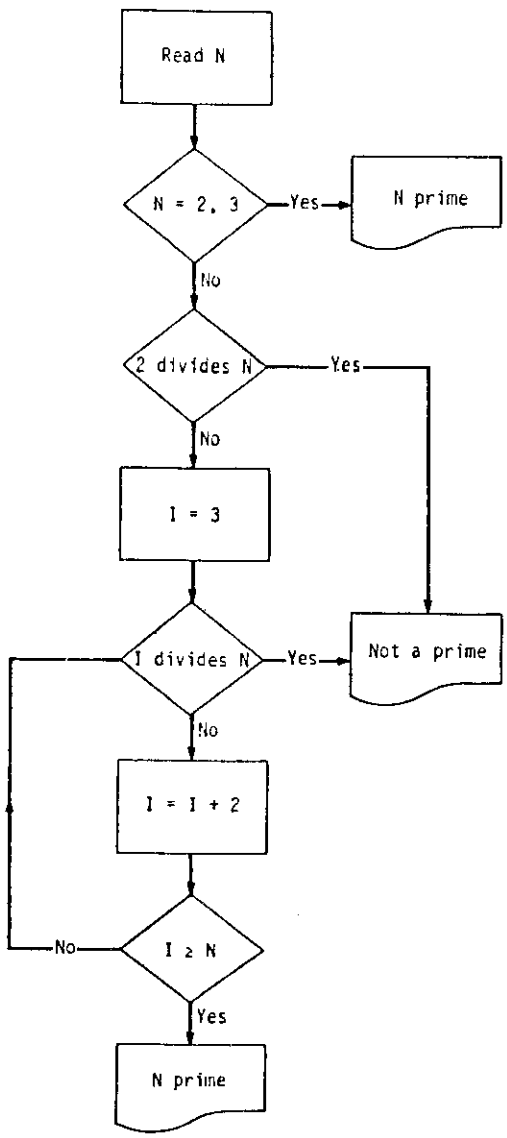

Figure Algorithm 2

จากอัลกอริทึมแบบที่ 2 จะเห็นได้ว่าประสิทธิภาพในการทำงานจะดีกว่าแบบแรก แต่ถ้าเราวิเคราะห์รายละเอียดลงไปจะพบว่าเซตของเลขที่ใช้ในการหารนั้น จะเป็นเซตที่ประกอบด้วยเลข {2,3,5,7,9,11,13,...} ซึ่งในเซตดังกล่าวก็จะมีตัวเลขบางจำนวนมี common factorเดียวกันเช่น 3,9,15,...เป็นต้น ดังนั้นเราจะพัฒนาอัลกอริทึมต่อไปโดยการกำหนดว่าตัวที่จะนำมาหารจำนวน N นี้จะต้องมีคุณสมบัติดังนี้คือ เลขดังกล่าวเริ่มจากค่า 2 และเป็นค่า prime number และจะต้องมีค่าน้อยกว่ารากที่ 2 ของ N จากเงื่อนไขนี้เราจะได้อัลกอริทึมรูปแบบที่ 3 ดังนี้

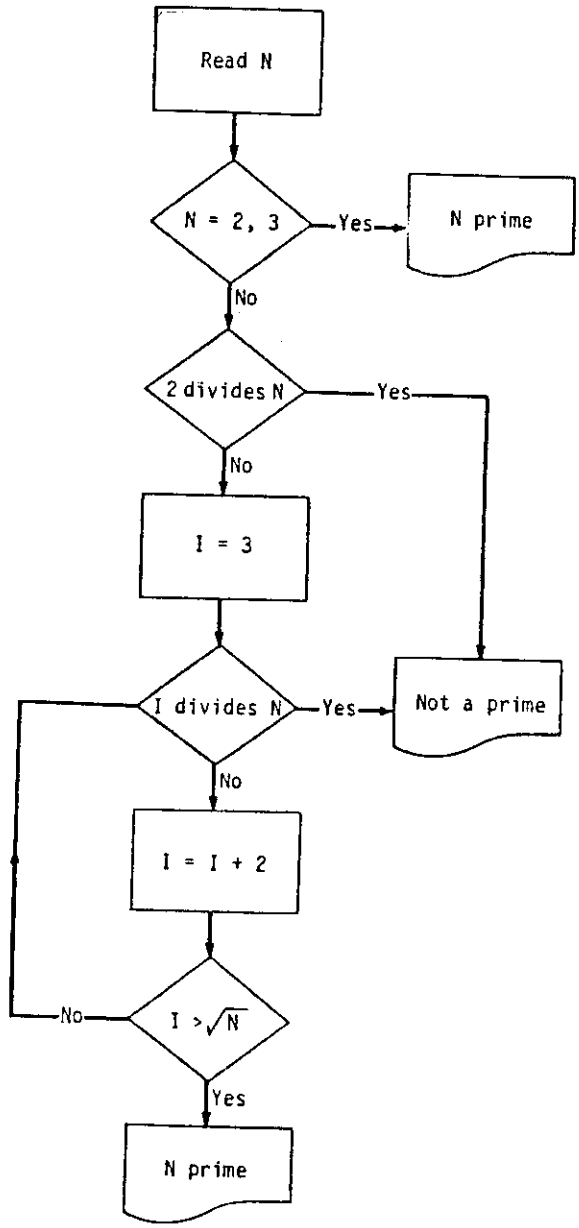

Figure . Algorithm 3

ตารางเปรียบเทียบอัลกอริทึมทั้ง 3 แบบประกอบกับค่า N ที่ต่างๆกันว่าแต่ละแบบจะมีประสิทธิภาพแตกต่างกันอย่างไร

N	ALGORITHM ALL I	ALGORITHM ODD I< N	ALGORITHM 2 AND ODD I<= N
10	8	5	2
100	98	50	5
1000	998	500	16

จากตารางจะเห็นได้ว่าถ้าค่าของ N ยิ่งมากจะยิ่งเห็นความแตกต่างของอัลกอริทึมแบบที่ 1 กับแบบที่ 3 ยิ่งชัดเจนขึ้นเช่นถ้า N มีค่าเป็น 1000 อัลกอริทึมแบบที่ 1 จะทำงานเป็น 60 เท่าของอัลกอริทึมแบบที่ 3 ในกรณีของการแก้ปัญหาเพื่อตอบคำถามว่า N เป็น prime numberหรือไม่นั้นเราจะตัดสินใจเลือกอัลกอริทึมแบบที่3ได้เลย แต่ถ้าเกิดว่าปัญหาที่จะให้ดำเนินการนั้นไม่ใช่ลักษณะดังกล่าวแต่กลับเป็นว่าให้เราหาว่าตัวเลขจำนวนเต็มที่มีค่าน้อยกว่า N มีตัวเลขใดบ้างที่เป็น prime number ซึ่งถ้าเป็น เช่นนี้แล้วการเลือกอัลกอริทึมเพื่อแก้ปัญหาในการที่จะต้องเปลี่ยนแปลงไปจากเดิม จะใช้อัลกอริทึมรูปแบบเดิมเลยไม่ได้

สรุปปัญหาของการเลือกอัลกอริทึมที่ดีนั้นจะมีกฎเกณฑ์ดังนี้คือ

- **หนึ่ง** อย่าเพิ่งสรุปลงไปว่าอัลกอริทึมที่อยู่ในใจเรานั้นจะเป็นอัลกอริทึมที่มีประสิทธิภาพที่สุดให้เราลองวิเคราะห์ว่านอกจากอัลกอริทึมที่เราคิดแล้วยังมีอัลกอริทึมอื่นอีกที่สามารถแก้ไขปัญหาของเราอีกกี่แบบบ้าง

- **สอง** ให้นำอัลกอริทึมแต่ละแบบที่ปรากฏมาวิเคราะห์แยกรายละเอียดถึงประสิทธิภาพแล้วนำมาเปรียบเทียบ แล้วจึงสรุปว่าควรจะเลือกอัลกอริทึมแบบไหนที่จะดีที่สุด

จะเห็นได้ว่าคำกล่าวที่ว่า " good algorithm is necessary but not a sufficient condition for a good program " ซึ่งแปลได้ว่าการมีอัลกอริทึมที่ดี เป็นเรื่องจำเป็นสำหรับการสร้างโปรแกรมที่ดีแต่เงื่อนไขแค่อัลกอริทึมเพียงประการเดียวก็ไม่เพียงพอที่จะทำให้โปรแกรมนั้นดีได้ ยังมีองค์ประกอบอย่างอื่นที่ประกอบด้วยหลักการที่สำคัญเหนือสิ่งอื่นใดก็คือผู้เขียนโปรแกรมควรที่จะเข้าใจปัญหาที่จะดำเนินการอย่างถ่องแท้ในการที่เลือกจะอัลกอริทึมที่ถูกต้องและชัดเจนในดำเนินการ ดังนั้นเพื่อให้นักศึกษาได้เข้าใจและมีประสบการณ์ในเรื่องของสร้างอัลกอริทึมดี ขอให้ศึกษาจากตัวอย่างต่อไปนี้

ตัวอย่างที่ 1 จงสร้างอัลกอริทึมเพื่อคำนวณหาพื้นที่ของสามเหลี่ยมโดยที่กำหนดตามทั้งสามของสามเหลี่ยมคือ A,B,C มาให้ตัวอย่างในข้อนี้นักศึกษาจึงต้องวิเคราะห์ก่อนว่าการใช้ สูตรเพื่อคำนวณหาพื้นที่สามเหลี่ยมจากสูตร

$$AREA = (S \star (S - A) \star (S - B) \star (S - C))^{1/2}$$

โดยที่ S มีค่าเท่ากับ $\frac{1}{2} \star (A + B + C)$จากสูตรการหาพื้นที่นี้นั้นจะเห็นได้

สูตรดังกล่าวจะใช้ได้ถ้าหากว่า

$$- (S - A) <= 0$$

$$- (S - B) <= 0$$

$$- (S - C) <= 0$$

โดยที่เงื่อนไขทั้ง 3 นี้จะมีเงื่อนไขใดเงื่อนหนึ่งเป็นจริงไม่ได้ดังนั้นบทสรุปของอัลกอริทึมในการคำนวณหาพื้นที่จึงออกมาในอัลกอริทึม ดังนี้คือ

$$S = \frac{1}{2} \star (A + B + C)$$

```
if      (S - A) <= 0
   or   (S - B) <= 0
   or   (S - C) <= 0
then    AREA      = 0
else    AREA      = ( S * (S - A) * (S - B) * (S - C))^1/2
end if
```

ตัวอย่างที่ 1 จงสร้างอัลกอริทึมสำหรับคำนวณหาผลรวมของเลขจำนวนหนึ่งที่รับจากแป้นพิมพ์ ในการเลือกอัลกอริทึมสำหรับตัวอย่างนี้ เราอาจจะกำหนดรูปแบบของอัลกอริทึมได้หลายวิธี แต่วิธีที่มักปฏิบัติและใช้กันก็คือ

แบบที่ 1 กำหนดให้ป้อนข้อมูลเพื่อบอกจำนวนตัวเลขจำนวนทั้งหมดที่จะรับว่ามีการหาผลบวกของ....จำนวนแล้วจึงใช้ While loop ควบคุม work flow ซึ่งจะได้ผลสรุปของอัลกอริทึมออกมาดังนี้

```
Total
read  count from keyboard
set total  = 0
while         count  >  0
        read (A)
        total   =   total + A
        count   =   count - 1
end of while
print  total
stop
```

จากตัวอย่างแบบที่ 1 ของการใช้อัลกอริทึมจะเห็นได้ว่าเราใช้ count เป็นตัวควบคุมใน while loop โดยมี count นั้นเราจะรับจากแป้นพิมพ์วิธีนี้ดีในแง่ที่ว่าเราสามารถจะใช้อัลกอริทึมนี้ในการหาเลขจำนวนเท่าใดก็ได้เพียงแต่ป้อนข้อมูลให้ ตัวแปร count ได้ถูกต้องแต่อัลกอริทึมนี้ก็ยังมีจุดบกพร่องอยู่ตรงที่ว่าถ้าหากเราจะต้องบวกเลข 20 จำนวนเราก็ต้องมานับว่ามีกี่จำนวนถ้านับผิดเป็นมากไปหรือน้อยไปก็ทำให้โปรแกรมทำงานผิดพลาดไปด้วยจากอัลกอริทึมแบบที่ 1 นี้ถ้าเราจะปรับแก้ใหม่ให้เป็น อัลกอริทึม แบบที่ 2 จะได้รูปแบบดังนี้

แบบที่ 1 เป็นแบบที่เราใช้ last data เป็นตัวตรวจสอบให้จบโปรแกรมโดยการกำหนดเงื่อนไขให้เลขจำนวนที่จะหาผลบวกใน total นั้นไม่มีกรณีเช่นนี้ซึ่งข้อกำหนดนี้ต้อง ระวังไม่ให้เป็นเงื่อนไขผูกให้อัลกอริทึมติดกับลักษณะของข้อมูลมากจนเกินไปดังนั้นถ้า

เราหากทราบว่าเลขจำนวนใดที่จะนำมาหาค่า total นั้นไม่มีสิทธิติดลบเราก็อาจจะนำมาใช้เป็นเงื่อนไขทดสอบได้ ผลสรุปของ อัลกอริทึม วิธีนี้จะได้รูปแบบดังนี้

```
total
        set  total  =  0
        read (A)
    while (A >= 0)
        total  =  total + A
        read (A)
    end of while
    print total
    stop
```

ผลสรุปของอัลกอริทึมวิธีนี้จะได้รูปดังนี้

```
total
    set  total  =  0
    read (A)
    while (A >= 0)
    total  =  total + A
    read (A)
    end of while
    print total
    stop
```

อัลกอริทึมแบบที่ 2 นี้มีข้อจำกัดคือค่าของ A จะน้อยกว่า 0 ไม่ได้ดังนั้นถ้าในการปฏิบัติงานจริงแล้วเกิดพบว่ามีเลขจำนวนใดจำนวนหนึ่งมีค่าน้อยกว่า 0 แล้วอัลกอริทึมนี้จะไม่สามารถหาค่า total ได้ตามต้องการอัลกอริทึมทั้งสองแบบนี้นำมาเปรียบเทียบให้พิจารณาดูนั้นจะขึ้นอยู่กับลักษณะงานแต่ละประเภทซึ่งผู้ใช้จะต้องพิจารณาว่าแบบใดเหมาะสมกับงานของตนเองที่สุด

ตัวอย่างที่ 4 จงสร้าง algorithm ในการรับข้อมูลมาเก็บใน array หรือ table ที่ชื่อว่า NUMBER แล้วทำการหาว่า element ใน array NUMBER มีค่าจำนวนใดที่เกินกว่า 100 อยู่กี่จำนวน

```
Examine Array
tally= 0.
index = 1.
read (lastcont)
while (index <= lastcount)
     read (Number(index))
     if Number(index) > 100
          tally = tally + 1
      endif
      index = index + 1
end of while.
```

ตัวอย่างที่ 5 จงเขียน algorithm เพื่อแบ่งจำนวนเงิน NET_PAY ให้เป็นธนบัตรและเหรียญชนิดต่างๆกันตามรูปแบบดังนี้

ธนบัตร หรือ เหรียญชนิด

1000	500	100	50	20	10	5	1

การกำหนดเงื่อนไขของ Input นั้นแล้วแต่สภาพที่เหมาะสมในทางปฏิบัติ เช่น อ่าน ข้อมูลจากแฟ้มข้อมูลแต่ในขั้นนี้จะสมมุติว่าเราจะรับข้อมูลของคนงานแต่ละคนซึ่งประกอบไปด้วย ชื่อ และ เงินเดือนจากแป้นพิมพ์โดยกำหนดกติกาว่าชื่อของคนงานนั้นเป็น string ยาว 7 ตัวอักขระและการรับข้อมูลจะสิ้นสุดลงถ้าพบว่าชื่อของคนงานที่รับเข้ามาเป็น "*******" ตัวแบบในงานนี้ที่เหมาะสมในการปฏิบัติงานก็คือกำหนดให้มี array DENOMINATION และ COUNTER มีขนาด 8 เซล โดยมีโครงรูปดังนี้.

ตัวอย่างที่ 3 จงสร้าง algorithm เพื่อจะคำนวณหาภาษีของคนงานจากยอดเงินที่จะจ่าย โดยกำหนดเงื่อนไขในการคิดภาษีจะทำตามเงื่อนไขจำนวนบุตรดังนี้

Gross wage $	Fewer than tree dependant	Tree or more dependant
100 or less	tax = null	tax = null
100.01 - 250	tax = 15 % of gross	tax = 10 % of gross
over 250	tax = 35 % of gross	tax = 25 % of gross

```
tax calculateion
      if Grosspay <= $ 100
        taxrate = 0
      else
        if Grosspay <= $ 250
             if dependent < 3
                  taxrate = 15
             else
                  taxrate = 10
             endif
        else
             if dependent < 3
                  taxrate = 35
             else
                  taxrate = 25
             endif
        endif
   endif
   tax = Grosspay * taxrate/100.
```

DENOMINATION	1000	500	100	50	20	10	5	1
COUNTER	0	0	0	0	0	0	0	0
	1	2	3	4	5	6	7	8

โดยที่ DENOMINATION จะทำการเก็บค่าของธนบัตรหรือเหรียญชนิดต่างๆ ส่วน COUNTER จะทำการเก็บผลที่ได้จากการคำนวณหาจำนวนธนบัตรหรือเหรียญชนิดต่างๆ ดังตัวอย่าง จากข้อมูลดังนี้

		1000	500	100	50	20	10	5	1
SALARY	8938	8	1	4	-	1	1	1	3

algorithm ที่ใช้ในการทำงานคือ

```
read (name)
while (name <> "*******")
      do
        readln (salary)
          begin
            remainder = salary
            index = 1
              while remainder > 0
                 do
                   begin
                     divide remainder by denomination(inddex)
                      placing quoitent in counter(index)
                      and remainder set index increment by 1
                end of while {remainder > 0}
           read(name)
      end of while {name <> "*******"}
```

แบบฝึกหัด

1. Each of the followinmg represents an attempt to write an algorithm that computer the sum of the first 10 integers. State why each is either poor or invaled.

(a) START

set sum to 1+2+3+4+5+6+7+8+9+10

write sum

END OF THE ALGORIOTHM

(b) START

set n to 10

set sum to 0

add n to the sum

decrement n by 1

set if n › 0

if it is go back and repeat the previous step

write sum

END OF THE ALGORIOTHM

(c) START

set i to 10

set sum to 0

repeat 10 times

add i to sum

end of the repeat loop

write sum

END OF THE ALGORITHM

(d) START

```
set i to 1
set sum to 0
while i < 10 do
    add 1 to sum
    increment i by 1
end of the while loop
write sum
```

END OF THE ALGORITHM

2. Write a valid algorithm to find the sum of the first k integers 1,2,...,k. the value of k will be external input to the algorithm. (Be sure to handle the ilegal situation of K <= 0)

3. Write an algorithm that ditermines the mode of a list mode is the value that occurred most frequently. For example, in

1 2 2 3 3 3 4 5 5 6 6 7 7 7 7

the mode is 7 and has a frequency of 4. You may assume that

(a) The values are has already sorted into ascending sequence.

(b) If two or more values occur an equal number of times, the mode will be defind as the numerically largest value. The algorithm you develop should not store all the data in a list; it should read

in only one value at a time and process that value. The output of your algorithm should be the mode and its frequency.

4. Develop an algorithm to solve quadratic equations of the form $ax^2 + bx + c = 0$ using the quadratic formula

$$\text{Roots} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Your algrithm should handle the regular cases as well as the special cases of;

(a) A double root ($b^2 - 4ac < 0$).
(b) Complex roots ($b^2 - 4ac < 0$).
(c) A nonquadratic equation ($a = 0$).
(d) An illegal equation ($a = 0, b = 0$).

5. Assume that you are given a string of input text of arbitary length and ending with a "$". Furthermore, assume that you wish to print this text as a set of line of width no greater than k columns (k will be external input to the algorithm). The rules of our printing also specify that we must never break up a word between lines (Assume the length of one word) will never exceed k) If a word cannot fit on the current line , print the current line, print the current line as is and go to on to the next line For example , if

k = 20 and the input text was: Hello. this an example of how textual material might look

the output would look like:

"Hello. This is an example of how some textual material migth look..."

6. Modify the algorithm of probelm 5 so that addition to printing line, it will *align* the left and right marging blnks between the word. If the actual lenght of the current line is n character, we need to add (k - n) blanks. We should add them as uniformly as posible between all words in the line (There is a possible ambigguity in this question if there is only one word on the line. Dicide what to do in case.) The output in problem 5 would now look like this.

"Hello. This is an example of how some textual material might look.."

7. (a) Assume that we have two lists that are sorted into ascending order. List A contains m items and list B contains n items. The values of m and n are not necessarily identical . Write an algorithm that merges all item of A and B into a single sorted list C containing all the elements of A and B.

(b) Modify the algorithm from part a so that it eliminates any duplicate item that appears in both lists A and B. Assume that lists A an B itself dose not contain any duplicates.

(c) Modify the algorithm from part a so that it eliments any duplicate items appearing in both lists A and B or that are duplicated withinlist A or list B.

(d) What is the time complexity of the merge operation in parts a,b, and c ?

8. (a) An alternative type of sort is called the *insertion sort*.if we are sorting a list A into ascending sequence we find the smallest item, place it in the first position of an entirely new list, B, and set the value in A in to infinity (or some very large value).Then we find the smallest value left in A move it to B, and set it to infinity. When we are done, B is sorted and A is destroyed. Write an algorithm to implement the insertion sort as described.

(b) What is this time complexity of this algorithm ? How much space dose it requie compared to the other algorithms described in this chapter?

9. Write an algorithm to input an integer an integer value N and determine if N is prime.(A prime number is an integer N >= 1 that is not evenly divisible by divisble by any values other than 1 and N) the output of your algorithm should be either: ' N ' is prime or ' N ' is not prime,' M ' is a factor . Make sure that your algorithm operaters properly for *all* values of N.

10. Assume that we represented a deck of cards by the integers 1, ... , 52 . using the following assignment.

1 = A	14 = A	27 = A	40 = A
2 = 2	15 = 2	28 = 2	41 = 2
.	.	.	.
.	.	.	.
.	.	.	.
13 = K	26 = K	39 = K	52 = K

Write an algorithm that will input 5 cards, corresponding to a 5-card poker hand, and determine if we have:

(a)A straight- any 5 cards in sequence, regardless of suit ; for example,

A-2-3-4-5, 9-10-J-Q-K.

(b)A flush- any 5 cards of the same suit, regardless of rank ; for example.

5 - 7 - 9 - 10 - K .

(c) A straight flush- 5 cards of the same suit in sequence; for example.

2 - 3 - 4 - 5 - 6 .

The output of the algorithm should indicate whether we had any of the three types of hands. (You can extend the problem to include other poker hands.) After writing the algorithm, comment on the data representation for the playing cards. Would you have done it differently? Would it have affected your algorithm?

11. Assume that we have N cities, numbered 1, 2,..., N, and a mileage chart giving the exact distance between any two cities directly connected by a railroad line. (If two cities are not directly connected, the mileage chart contains a o.) For example, if N = 4 and the railroad connections were

FROM	1	2	3	4
1	-			
2	0			
3	100	70	50	
4	0	0	50	-

(The chart assumes that trains can always go in either direction on a track.)Develop an algorithm that will input any two indices i,j (1<= i, j<=N; i<j) and determine the total rail mileage from city j. For example, if the input were (********)
the algorithm should output 150 miles (Be careful of the pathological case of two cities that are not connected by rail.)

12. Assume that we have two lists, x_i and y_i, both containing n items. The rank correlation coeffcient, r, between the lists x and y is defined as follows.

$$r = 1 - \frac{6 * \sum_{i=1}^{n} (a_i - b_i)^2}{n * (n^2 - 1)}$$

where a_i and b_i are the ordinal rankings of the raw scores contained in x and y, respectively. For example, if n = 4 and x and y are

$x_1 = 52$ $y_1 = 17$

$x_2 = 30$ $y_2 = 93$

$x_3 = 79$ $y_3 = 62$

$x_4 = 60$ $y_4 = 77$

then a and b are

$a_1 = 3$ $b_1 = 4$

$a_2 = 4$ $b_2 = 1$

$a_3 = 1$ $b_3 = 3$

$a_4 = 2$ $b_4 = 2$

Assume also that there exists an algorithm called: Rank(x,a) that takes a set of raw scores in x , determines the proper rankings , and stores them in a . Rank(y,b) will do the same for y and b ,respectively .If Rank encounters a raw score of -1 , implying a missed examination , it assigna a rank of -1. Without worring about how Rank is implemented, Write an algorithm to solve the problem whose specifications are contains in figure 11

13. Develop the algorithm Rank that was used in the correlation assignment in exercise 12 . Rank should assign to a raw score of -1 a ranking of -1 and should handle ties by assigning the average of all the ranks to the scores.