سایت ویژه ریاضیات www.riazisara.ir

درســـنامه ها و جـــزوه های ریاضـــی

سوالات و پاسخنامه تشریحی کنکور

نمونه سوالات امتحانات ریاضی

نرم افزارهای ریاضیات

و...

فصل دهم
الگوریتم ها
Algorithms
بخش اول
متغیر ها و مقدار دهی

این فصل را به تئوری اعداد اختصاص داده بودم. همه چیز آماده نگارش تئوری اعداد بود ، که تصادفا کتاب ارزشمند **نظریه اعداد تالیف روانشاد خانم مریم میرزا خانی** بدستم رسید. دیدم اگر به نوشتن فصل تئوری اعداد ادامه دهم ، زیر ه به کرمان بردن است. لذا از این کار منصرف شدم. توصیه می شود خوانندگان محترم این کتاب ارزشمند را مطالعه کنند. می گویند نظریه اعداد ملکه ریاضیات است. کتاب خانم میرزا خانی ملکه کتب ریاضی به زبان فارسی است. روانش شاد و یادش گرامی.

مقدمه - کلمه الگوریتم از الخوارزمی دانشمند ریاضیات ایرانی گرفته شده است و گویا ایشان اولین مخترع الگوریتم هستند.

الگوریتم در علوم کامپیوتری و ریاضیات گسسته اهمیت زیادی دارد.
الگوریتم عبارت است از یک رشته دستور ها ، که اگر بکار برده شود ، نتیجه مطلوب را می دهد. مثلا دستورا لعمل پختن کیک ، یک الگورتم است. اگر دستورات را عمل کنید ، یک کیک بدست می آورید.
یک الگوریتم دارای **ورودی Input** است.
ورودی عبارت است از مواد و یا اطلاعاتی که الگوریتم بکار می برد.
الگوریتم یک **خروجی Output** دارد
خروجی عبارت است از نتیجه پایانی الگوریتم.
در مورد دستور العمل کیک ، مواد لازم ، ورودی است. دستور العمل، الگوریتم است. و خروجی، کیک است.
اجرای Run or Execute یک الگوریتم عبارت است از بکار بردن ورودی برای بدست اودن خروجی
امروزه کلمه الگوریتم به سلسله مراحلی گفته می شود که به یک زبان کامپیوتری نوشته می شود و بوسیله کامپیوتر اجرا می شود. ورودی و خروجی هم اطلاعاتی است که به کامپیوتر می دهیم و از کامپیوتر می گیریم.
هنگام جستجو در گوگل ، سبب می شویم که یک الگوریتم اجرا شود. الگوریتم گوگل یک کلمه و یا یک عبارت را به عنوان ورودی می گیرد و یک لیست از صفحات وب به عنوان خروجی به ما می دهد.
هنگامی که کلید $Enter$ را روی صفحه کلید فشار می دهیم ، سبب می شود که الگوریتم گوگل اجرا شود، ودرنتیجه خروجی روی صفحه به ما نشان داده می شود.
انجام این کار برای ما بدون زحمت است. اما یک گروه از متخصصان ماهر آنرا طرح و اجرا کرده اند.
این فصل مقدمه ای است برای آشنایی به این مهارت ها است.

گر چه بحث ما مقدماتی است ، اما آنچه اینجا ارائه می دهیم ، اگر بیشتر مورد استفاده قرار گیرد ، می تواند برای الگوریتم های پیچیده و مهم بکار گرفته شود.
برای این که کار را ساده کنیم ، روی الگوریتم هایی متمرکز می شویم که بطور ساده با اطلاعات ورودی شروع می شوند ، روی آنها کار می شود ، ، در نهایت منتج به خروج اطلاعات می شود.
برای سادگی کار ، اطلاعات ورودی و خروجی بیشتر عددی یا الفبا عددی است. اما تصور نشود که این کار سبب محدود شدن فعالیت ها بشود. هر الگورتیم پیچیده را می توان به الگورتیم های ساده تر تجزیه کرد.
نکته آخر این که روی یک زبان کامپیوتری مشخصی تمرکز نمی کنیم. بلکه شبه برنامه های Pseudocodes ارائه می دهیم که برای هر زبان کامپیوتری قابل استفاده است. تمام الگوریتم ها یی که ارائه می دهیم قابل استفاده برای تمام زبان های کامپیوتر ی است.

۱.۵ ۱ متغیر ها و دستورمقدار دهی Variables and the Assignment Command

در یک الگوریتم ، یک متغیر A Variable یک نماد است که می توان به آن مقادیر مختلفی اختصاص داد. مانند آنچه در جبر دیدیم ، حروف $a, b, , c, \cdots, z$ متغیر ها هستند. اگر بخواهیم می توان آنها را اندیس دار کرد. مثلا $x_۱, x_۲, x_۳$ سه متغیرمختلف هستند.

اشکالی ندارد که بجای حروف ، یک نام به عنوان متغیر بکار بریم. در زبان کامپیوتری ، یک متغیر نمایش یک جایی در حافظه کامپیوتر است. متغیر می تواند در زمانهای مختلف مقادیر مختلفی داشته باشد ,اما در هر زمان فقط یک مقدار می تواند داشته باشد. هنگامی که الگوریتم به جریان می افتد ، یک متغیر می تواند درزمان های مختلف ، مقادیر مختلفی داشته باشد.

یک الگوریتم عبارت است از یک سلسله از دستوره ها یا فرمان ها Instructions or Commands دستوری که می گوید به متغیر x مقدار ۲ داده شود ، به صورت زیر بیان می شود.

$$x := ۲$$

که می خوانیم به x مقدار ۲ اختصاص داده شده است و یا x عدد ۲ می گیرد.
هنگامی که این دستور اجرا می شود ، x نماینده عدد ۲ ، است تا وقتی که مقدار دیگری به آن داده شود. اگر بعداً دستور زیر صادر شود

$$x := ۷$$

پس x بجای عدد ۷ است. در دستور بعدی

$$y := ۲ * x + ۱$$

پس متغیر y بجای عدد ۱۵ است و یا به عبارت دیگر مقدار ۱۵ به y داده شده است. پس y بجای عدد ۱۵ عمل می کند. اگر دستور بعدی

$$y := y + ۲$$

باشد ، پس هنگامی که آن دستور اجرا شود ، y مقدار $۱۷ = ۲ + ۱۵$ خواهد داشت.

کلمه متغیر در الگوریتم ، کمی با مفهوم متغیر در جبر متفاوت است. در الگوریتم ، یک متغیر یک مقدار معینی در یک زمان مشخص دارد . اما ، در جبر، یک متغیر احتمالا یک مقدار نا متناهی است. در الگوریتم عبارت

$$y := y + ۲$$

یعنی y یک مقدار جدید می گیرد. به عبارت دیگر ، مقدار قبلی آن به اضافه ۲
بر عکس در جبر معادله $y = y + ۲$ هیچ پاسخی ندارد.
در الگوریتم ، $y :=: ۲$ و $y = ۲$ با هم متفاوت هستند. در الگوریتم عبارت $y = ۲$ یک جمله باز است که می تواند صحیح یا غلط باشد.

مثلا اگر در یک الگوریتم ، دستور $y := ۲$ صادر کردیم. پس عبارت $y = ۲$ مقدار T دارد یعنی صحیح و $y = ۳$ مقدار F دارد یعنی غلط. در فصل منطق به مفاهیم T و F اشاره کردیم.
T حرف اول $True$ یعنی صحیح و F حرف اول $False$ یعنی غلط است.
همچنین مقدار y هر چه باشد، $y = y + ۲$ غلط است ،

۱۰.۲ - حلقه ها و مفهوم الگوریتم Loops and Algorithm Notation

زبان های کامپیوتری انواع حلقه ها بکار می برند. این حلقه ها یک سری دستور هایی را چندین مرتبه اجرا می کنند. یکی از اساسی ترین حلقه ها **حلقه مادامی که** **While Loop** نامیده می شوید.
$While$ یک کلمه کلیدی است که یک سری دستور ها یی را مادامی که $while$ یک جمله باز $P(x)$ صحیح باشد، انجام می دهد. x یک متغیر است.
حلقه مادامی که دارای ساختمان زیر است.

با کلمه **while** شروع می شود و با کلمه **end** ختم می شود. این دو کلمه یک سری دستور هایی را دربر می گیرند. خط عمودی که بین $while$ و end ملاحظه می کنید ، بخاطر شما می آورد که دستور ها $Command$ همه با هم داخل حلقه مادامی که $while\ loop$ قرار دارند.

while $P(x)$ **do**
| Command 1
| Command 2
| ⋮
| Command n
end

ترجمه حلقه بالا
مادامی که $P(x)$ **انجام بده**
دستور اول
دستور دوم
دستور سوم
⋮
دستور n ام
تمام

وقتی که $while\ loop$ شروع به کار می کند ، متغیر x یک مقدار معینی دارد. اگر $P(x)$ صحیح باشد ، $while\ loop$ دستور اول تا n ام را انجام می دهد. ممکن است در دستور n ام مقدار x تغییر کند. اگر $P(x)$ هنوز صحیح باشد ، دستور اول تا n ام مجددا اجرا می شود. این کار ادامه پیدا می کند تا $P(x)$ غلط بشود. اینجا $loop$ تمام می شود و الگوریتم به دستور بعد از $loop$ حرکت می کند.

اولین مرتبه که $while\ loop$ سری اول دستور ها را اجرا می کند
به آن **اولین تکرار** **First Iteration** می گویند.
مرتبه دوم اجرای دستورها را **تکرار دوم** **Second Iteration** می گویند ، الی آخر.

وقتی که $while\ loop$ شروع می شود ، اگراتفاقا $P(x)$ قبلا غلط باشد ، $while\ loop$ کاری نمی کند .

توجه :کلمات کلیدی Key Words کلماتی هستند که فقط مخصوص همان زبان کمپیوتری است و بر نامه نویس نمی تواند آنرا بجز در مواردی که آن زبان معین کرده ، بکار برد.

اجازه دهید به چند مثال نگاه کنیم.این مثال ها، دستور **output** را اجرا می کنند. یعنی خروجی $while\ loop$ مقدار y است.

مثال اول

$x := 1$
while $x \leq 6$ **do**
  $y := 2 \cdot x$
  output y
  $x := x + 1$
end

iteration	1	2	3	4	5	6
x	2	3	4	5	6	7
y	2	4	6	8	10	12

اولین خط به x مقدار ۱ داده ایم. یعنی $x := ۱$ فراموش نشود $x := ۱$ با $x = ۱$ تفاوت دارد. تفاوت را در بخش اول توضیح دادیم.
کامپیوتر گزاره $x \leq ۶$ را چک می کند. چون مقدار $x \leq ۶$ است ، پس گزأره صحیح است. دستور $y := ۲ * x$ اجرا می شود. مقدار y می شود ۲ حالا مقدار ۲ به خروجی می رود. حالا دستور $x := x + ۱$ اجرا می شود .حالا مقدار $x = ۲$ است. در تکرار دوم باز مقدار x که ۲ است چک می شود و چون گزاره صحیح است ، پس دستور ها مجددا تکرار می شوند. در تکرار ششم $x = ۷$ است، پس $x > ۶$ است که دیگر صحیح نیست ، لذا $loop$ دیگر تکرار نمی شود. از روی جدول ملاحظه می کنید که خروجی ۲, ۴, ۶, ۸, ۱۰, ۱۲ است.

حالا یک تغییر جزئی در مثال اول ایجاد می کنیم. به این صورت که دستور **output** یعنی **خروجی** را بعد از **end** قرار می دهیم. این مرتبه هیچ خروجی نداریم تا $while\ loop$ تمام شود. بعد از اتمام $loop$ خروجی $y = ۱۲$ است.

مثال دوم

$x := 1$
while $x \leq 6$ **do**
  $y := 2 \cdot x$
  $x := x + 1$
end
output y

حالا به مثال سوم توجه کنید. همه چیز مانند مثال دوم است ، با این تفاوت که بجای $x := x + 1$ می نویسیم $x := x - 1$ پس مقدار x در هر تکرار ، کوچک تر می شود. و برای همیشه مقدار $x \leq 6$ است ، که صحیح است. پس $loop$ برای همیشه تکرار می شود و هرگز متوقف نمی شود. به این حالت می گویند **حلقه نا محدود** **infinite loop**

مثال سوم

```
x := 1
while x ≤ 6 do
    y := 2 · x
    x := x − 1
end
output y
```

ما الگوریتم را یک مجموعه از دستورات تلقی می کنیم ، که یک کار را در دفعات محدود انجام می دهد. بنا بر این باید از حلقه های نا محدود احتراز کرد.

حالا با اطلاعاتی که تا کنون بدست آورده ایم ، می توانیم چند الگوریتم کامل بنویسیم. یک الگوریتم با یک عنوان شروع می شود. این عنوان با کلمه الگوریتم Algorithm شروع می شود. سپس شرح مختصری در مورد کاری که الگوریم انجام می دهد ، و بعد از این توضیح ، ورودی input و خروجی output شرح داده می شود. در نهایت متن اصلی body که شامل یک سری دستورات بین کلمات شروع begin و پایان end است ، می آید. معمولا برای وضوح بیشتر ، هر دستور را در یک خط می نویسیم. ممکن است توضیحات هر دستور را سمت راست آن بنیویسیم. بین توضیحات و دستور، چند نقطه می گذاریم تا مشخص شود آنچه بعد از نقطه چین می آید ، توضیح است. این توضیحات فقط برای فهم خواننده است. اگر توضیحات در کامپیوتر نوشته می شود ، کامپیوتر ، توضیحات را نا دیده می گیرد. البته هر زبانی برای خودش ، شرایطی برای نوشتن توضیحات دارد. باید آن شرایط را برای همان زبان رعایت کرد.

برای روشن شدن مطالب بالا یک مثال می آوریم. در این مثال ، ورودی یک عدد صحیح n است و خررجی ، n تای اول اعداد صحیح زوج مثبت است. مثلا اگر ورودی ۶ باشد ، خروجی لیست ۲, ۴, ۶, ۸, ۱۰, ۱۲ است.

Algorithm 1: computes the first n positive even integers

```
Input: A positive integer n                    (Tells reader what the
Output: The first n positive even integers      input & output is.)
begin
    x := 1
    while x ≤ n do
        y := 2 · x  ..... y is the xth even integer
        output y
        x := x + 1  ..... increase x by 1
    end
end
```

ترجمه الگوریتم بالا

الگوریتم ۱ :n تای اول اعداد صحیح زوج مثبت محاسبه می کند.
ورودی :یک عدد صحیح مثبت n به خواننده می گوید ورودی چیست.
خروجی :n تای اول اعداد صحیح زوج مثبت است به خواننده می گوید خروجی چیست.
شروع کن
$x := ۱$
مادامی که $x \leq n$ **انجام بده**
$y := ۲ * x$ متغیر y عدد صحیح زوج n ام است.
y **به خروجی بفرست**
$x := x + ۱$ به x یک عدد اضافه کن.
تمام
تمام

توجه :خطوط بنفش رنگ ، توضیحات است.

در بسیاری از زبان های کامپیوتری علاوه بر $while\ loop$ یک نوع حلقه دیگر بنام **حلقه انجام بده** دارد ، در ذیل ساختمان آنرا ملاحظه می کنید. اینجا i یک متغیر است ، m و n اعداد صحیح با $m \leq n$

for $i := m$ **to** n **do**
| Command
| Command
| ⋮
| Command
end

ترجمه الگوریتم بالا.
برای $i := m$ **تا** n **انجام بده**
دستور
دستور
⋮
دستور
تمام

در تکرار اول $for\ loop$ به i عدد صحیح m می دهد. یعنی $i := m$ و دستور های بین اولین و اخرین خط را انجام می دهد. در تکرار بعدی $i := m + ۱$ می شود یعنی به i یک عدد اضافه می شود. و دستور ها را دوباره اجرا می کند. سپس به i یکی اضافه می شود ، یعنی $i := m + ۲$ و دستور ها را انجام می دهد. الی آخر. در هر تکرار به i یک عدد اضافه می شود تا وقتی که $i := n$ بشود و در آخرین تکرار از حلقه بیرون می آید. هیچ کدام از دستورات مقادیر i, m, n تغییر نمی دهند. این کار را همان فرمان $do\ loop$ انجام می دهد.

جهت توضیح بیشتر ، اجازه دهید الگوریتم اول را با $for\ loop$ باز نویسی کنیم.

Algorithm 2: computes the first n positive even integers

```
Input: A positive integer n
Output: The first n positive even integers
begin
    for i := 1 to n do
        y := 2·i  ..... y is the ith even integer
        output y
    end
end
```

ترجمه الگوریتم بالا

الگوریتم ۲ :محاسبه n تای اول اعداد صحیح زوج مثبت
ورودی :یک عدد صحیح مثبت n
خروجی : n تای اول اعداد صحیح زوج مثبت
شروع کن
برای $i := ۱$ **تا** n **انجام بده**
$y := ۲ * i$ متغیر y عدد صحیح زوج i ام است.
y **را به خروجی بفرست**
تمام
تمام

۱.۳ عملگر های منطقی در الگوریتم ها Logical Operators in Algorithms

یک ارتباط تنگا تنگی بین الگوریتم ها و منطق وجود دارد. یک حلقه مادامی که *while loop* به انجام دستور ها ادامه می دهد ، تا زمانی که یک جمله بازصحیح است. این جمله باز ممکن است شامل چندین متغیربوده ، و خود از جمله های باز دیگر تشکیل شود که توسط عملگر های منطقی به هم متصل شده باشند . مثلا حلقه *loop* زیر ، تا زمانی که $P(x) \vee \sim Q(y)$ صحیح است ، لیست دستورات داده شده را اجرا می کند.

while $P(x) \vee \sim Q(y)$ **do**
| Command
| Command
| ⋮
end

امید است عملگر های منطقی را فراموش نکرده باشد. در صورت لزوم به فصل سوم مراجعه کنید. برای یاد آوری ، نماد $\vee$ به معنی یا است. و نماد $\sim$ نقیض است. $\sim Q$ یعنی نه Q

در الگوریتم بالا لیست دستور ها ، باید مقادیر x و y را تغییر دهند ، تا بالاخره $P(x) \vee \sim Q(y)$ غلط بشود و برنامه از *while do* خارج شود، در غیر این صورت ، حلقه نا متناهی خواهیم داشت.

طریق دیگری که الگوریتم ها می توانند عملگر های منطقی بکار برند ، ترکیب **if-then** است که به معنی **اگر – پس**

if $P(x)$ **then**
| Command
| Command
| ⋮
end

اگر $P(x)$ صحیح باشد ، پس الگوریتم لیست دستور ها ی بین **then** و **end** را اجرا می کند. اگر $P(x)$ غلط باشد ، کاری انجام نمی دهد و الگوریتم ، دستور های بعد از **end** را انجام می دهد.
نا گفتنه نماند که جمله ساده $P(x)$ می تواند جمله مرکب $P(x) \vee \sim Q(y)$ یا هر جمله مرکب دیگر باشد.

شکل دیگر دستور **if-then** دستور **if-then-else** است

```
if P(x) then
    Command
    Command
        ⋮
else
    Command
        ⋮
end
```

اگر $P(x)$ صحیح باشد ، الگوریتم ، اولین مجموعه دستورات بین **then** و **else** را اجرا می کند. اگر $P(x)$ غلط باشد ، دومین مجموعه دستورات بین **else** و **end** انجام می شود.
کلمه **else** یعنی در غیر این صورت.

اجازه دهید یک الگوریتم بنویسیم که ورودی آن n است و خروجی آن $n!$ است. پس الگوریم ما ساختمان زیر را خواهد داشت.

```
if n = 0 then
  | output 1   ..... because 0! = 1
else
  | Compute y := n! .....(we need to add the lines that do this)
  | output y
end
```

ترجمه بالا
اگر $n = 0$ **پس**
برون بده ۱ زیرا $0! = 1$ است.
در غیر این صورت
محاسبه کن $y := n!$ لازم است خطوطی را اضافه کنیم که این کار را انجام دهد.
به خروجی بفرست y
تمام

حالا باید خطوطی در الگوریتم بالا اضافه کنیم که $y = 1 * 2 * 3 * 4 * \cdots * n$ را محاسبه کند. برای این کار ، ابتدا $y = 1$ قرار می دهیم و سپس یک حلقه برای $for\ loop$ اضافه می کنیم که y را در ۱ سپس در ۲ سپس در ۳ و الی آخر ضرب کند. تا این ضرب کردن به n برسد.
الگوریتم کامل را در ذیل ملاحظه می کنید.

Algorithm 3: computes $n!$

```
Input: A non-negative integer n
Output: n!
begin
    if n = 0 then
        output 1 ..... because 0! = 1
    else
        y := 1
        for i := 1 to n do
            y := y · i
        end
        output y ..... because now y = n!
    end
end
```

ترجمه الگوریتم بالا

الگوریتم ۳ : محاسبه $n!$
ورودی : یک عدد صحیح نا منفی n
خروجی : $n!$
شروع کن
اگر $n = 0$ **پس**
به خروجی بفرست ۱ زیرا $0! =$ ۱ است.
در غیر این صورت
$y :=$ ۱
برای $i :=$ ۱ **تا** n **انجام بده**
$y := y *$ ۱
تمام
به خروجی بفرست y
تمام
تمام

در الگوریتم ها اغلب اطلاعات به صورت لیست هستند. یک لیست معمولا چندین عضو یا ورودی دارد. لذا هنگامی که یک لیست در حافظه کامپیوتر ذخیره می شود ، در یک مکان واحد ذخیره نمی‌شود ، بلکه در مکان های مختلف ذخیره می شوند. یک لیست دارای طول پنج مانند $X = (۲, ۴, ۷, ۴, ۳)$ ممکن است در شش مکان پشت سر هم ذخیره شود. اولین مکان مربوط به طول X است.

5	2	4	7	4	3
X	x_1	x_2	x_3	x_4	x_5

مکان X در حافظه کامپیوتر عدد ۵ است ، این عدد مشخص می کند که پنج مکان ذخیره شده بعدی مربوط است به ورودی های لیست X, اولین مکان بلا فاصله بعد از X با x_1 مشخص می کنیم ، بعدی x_2 است و الی آخر.

الگوریتم بعدی ، بزرگ ترین مقوله یا ورودی یک لیست را پیدا می کند. کلمه $biggest$ را که به معنی بزرگ ترین است به عنوان متغیر بکار می بریم. الگوریتم ، ابتدا اولین مقوله یا ورودی را بزرگ ترین فرض می کند. سپس در لیست حرکت می کند ، و بزرگ ترین را با هر مقوله ای که بزرگ تر است جابجا می کند. مثلا اگر لیست شامل $(۱, ۵, ۲, ۷, ۳, ۹)$ باشد ، ابتدا عدد ۱ یک را به عنوان بزرگ ترین عدد لیست انتخاب می کنند ، سپس به ۵ می رسد ، مقایسه می کند و چون $۱ < ۵$ است ، ۵ را بزرگ ترین به حساب می آورد. سپس به ۲ می رسد و مقایسه می کند ، اما چون $۵ < ۲$ است ، بدون تغییربزرگ ترین ، به رقم بعدی می رسد که ۷ است ، مقایسه می کند و چون $۵ < ۷$ است ، مقدار بزرگ ترین را عوض می کند و الی آخر.

Algorithm 4: finds the largest entry of a list

```
Input: A list X = (x1, x2, ..., xn)
Output: The largest entry in the list
begin
    biggest := x1 ..... this is the largest value found so far
    for i := 1 to n do
        if biggest < xi then
            biggest := xi ..... this is the largest value found so far
        end
    end
    output biggest
end
```

ترجمه الگوریتم بالا

الگوریتم ۴ : بزرگ ترین ورودی یک لیست را پیدا می کند.

ورودی : یک لیست $X = (x_1, x_2, \cdots, x_n)$

خروجی : بزرگ ترین ورودی لیست

شروع کن

$biggest := x_1$ این بزرگ ترین مقداری است که تا کنون پیدا شده.

برای $i := ۱$ **تا** n **انجام بده**

اگر $biggest < x_i$ **پس**

$biggest := x_i$ این بزرگ ترین مقداری است که تا کنون پیدا شده.

تمام

تمام

به خروجی بفرست $biggest$

تمام

حالا یک الگوریتم می نویسیم برای مرتب کردن لیستی از اعداد. مثلا اگر $X = (۴, ۵, ۱, ۲, ۱, ۳)$ باشد ، خروجی $(۱, ۱, ۲, ۳, ۴, ۵) = X$ خواهد بود.
مثال دیگر، یک لیست کاملا نا مرتب $(۵, ۴, ۳, ۲, ۱) = X$ را در نظر بگیرید. اولین و دومین نا مرتب هستند ، پس جای این دو را عوض می کنیم. لیست جدید می شود $(۴, ۵, ۳, ۲, ۱) = X$ این عمل در ردیف دوم جدول ۱۰.۳.۱ نشان داده شده است. کلمه $swap$ که در جدول ملاحظه می کنید به معنی عوض کردن یا جا بجا کردن است. سپس به ورودی دوم این لیست جدید X می رویم.این ورودی و ورودی سوم هم نا مرتب است ، پس آنها را جابجا می کنیم. حالا $(۴, ۳, ۵, ۲, ۱) = X$ است که در ردیف سوم جدول می بینید. به همین ترتیب ادامه می دهیم ، یعنی از چپ به راست حرکت می کنیم. در این لیست ، چهار جا بجا یی داشتیم. کلمه $1st\ pass$ که در ستون منتها آلیه سمت راست می بینید یعنی پاس اول یا دور اول

جدول ۱۰.۳.۱

5	4	3	2	1		
4	5	3	2	1	swap # $i=1$	
4	3	5	2	1	swap # $i=2$	
4	3	2	5	1	swap # $i=3$	$k=1$ (1st pass)
4	3	2	1	5	swap # $i=4$	

حالا آخرین ورودی یعنی ۵ در جای صحیح قرار دارد ، آخر لیست است. به این ترتیب ۵ را که ابتدای لیست بود ، به انتهای لیست بردیم. اما آنهایی که سمت چپ ۵ قرار دارند ، در جای صحیح نیستند.

به همین ترتیب ۴ را آنقدر جا بجا می کنیم تا قبل از ۵ قرار گیرد. جدول ۱۰.۳.۲

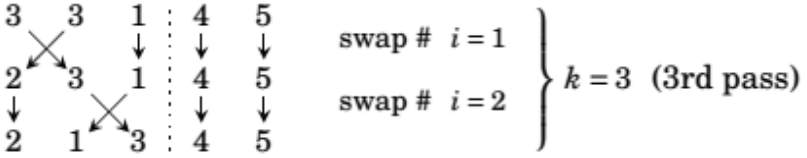
جدول ۱۰.۳.۲

4	3	2	1	5		
3	4	2	1	5	swap # $i=1$	
3	2	4	1	5	swap # $i=2$	$k=2$ (2n pass)
3	2	1	4	5	swap # $i=3$	

حالا دو قلم آخر در جای صحیح قرار دارند. این پاس دوم یا دور دوم بود.

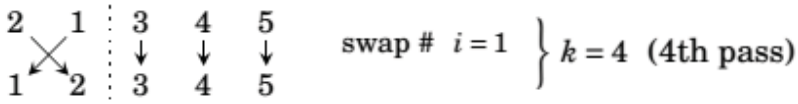
در پاس سوم عدد ۳ را آنقدر جابجا می کنیم تا در جای صحیح خود قرار گیرد. جدول ۱۰.۳.۳

جدول ۱۰.۳.۳



و در نهایت پاس چهارم داریم ، اینجا همه اعداد در جای صحیح قرار دارند و لیست ما از کوچک به بزرگ مرتب شده است. جدول ۱۰.۳.۴

جدول ۱۰.۳.۴



در الگوریتم بعدی این مراحل را انجام می دهد. توجه داشته باشید که در این لیست که کاملا نا مرتب بود مجبور شدیم دو تا دو تا جا بجا کنیم ، اما اگر یک ورودی در جای صحیح قرار داشته باشد ، آنرا جا بجا نمی کنیم .

در مثال مرتب کردن بالا طول لیست $n = ۵$ بود که مجبور بودیم $n - ۱ = ۵ - ۱ = ۴$ پاس انجام دهیم. این نوع مرتب کردن را **مرتب کردن حبابی Bubble Sort** می نامند. زیرا مانند حباب آب که از پایین به سطح آب می آید، عمل می کند. این نوع مرتب کردن خیلی ابتدایی است و برای یک لیست بلند و بالا خوب نیست. در ذیل الگوریتم کامل مرتب کردن حبابی ملاحظه می کنید.

Algorithm 5: (Bubble Sort) sorts a list

```
Input: A list X = (x_1, x_2, ..., x_n) of numbers
Output: The list sorted into numeric order
begin
    for k := 1 to n − 1 do
        for i := 1 to n − k do
            if x_i > x_{i+1} then
                temp := x_i ..... temporarily holds value of x_i
                x_i := x_{i+1}
                x_{i+1} := temp .....now x_i and x_{i+1} are swapped
            end
        end
    end
    output X ..... now X is sorted
end
```

همانطور که گفته شد ، دانشمندان کامپیوتر ، الگوریتم ۵ را **مرتب کردن حبابی Bubble Sort** می نامند ، زیرا عدد کوچک تر مثل حباب به ابتدای لیست می آید.

الگوریتم مرتب کردن حبابی دارای یک $for\ loop$ داخل یک $for\ loop$ دیگر است. در برنامه نویسی ، یک حلقه داخل حلقه دیگر را **حلقه تو در تو Nested Loop** می نامند. حلقه های تو در تو، در طراحی الگوریتم ها خیلی متداول است.

ممکن است متوجه شده باشید که الگوریتم ۵ یک عیب دارد. اگر مثلا $n = ۱$ باشد چه اتفاقی می افتد؟ اگر لیست ما $X = (۳)$ بود ، چه می شد؟ همین طور اگر لیست ما خالی بود. پس اولین $for\ loop$ سعی می کرد $for\ k := ۱\ to\ 0\ do$ را اجرا کند و این سبب می شد که یک حلقه نا متناهی ایجاد شود. برای رفع این مشکل ، می توانستیم یک $if\ else$ اضافه کنیم. اما هدف ما از الگوریتم ۵ صرفا معرفی مرتب کردن حبابی بود و نه یک برنامه قابل اجرا در کامپیوتر.

تمرینات۱۰.۳ — ۱۰.۲ — ۱۰.۱

۱ - خروجی الگوریتم زیر را پیدا کنید.

```
x := 1
y := 10
while x^2 < y do
    y := y + x
    x := x + 1
end
output x
output y
```

۲ - خروجی الگوریتم زیر را پیدا کنید

```
a := 0
b := 3
for i := 1 to 8 do
    if a < b then
        a := a + i
    else
        b := b + a
    end
end
output a
output b
```

۳ - طول یک لیست، یک عدد زوج است. اگر لیست $X=(3,5,8,4,6,8,7,4,2,3)$. باشد ، خروجی برای این لیست را پیدا کنید.

Algorithm

Input: $X=(x_1,x_2,\ldots,x_n)$
begin
  for $i:=1$ **to** $\frac{n}{2}$ **do**
    $k:=2i$
    $x_k:=x_k+1$
  end
  for $j:=1$ **to** $\frac{n}{2}$ **do**
    $k:=2j-1$
    $x_k:=x_k-1$
  end
  output X
end

۴ - **دنباله فیبو نا چی Fibonacci Sequence** دنباله $1,1,2,3,5,8,13,21,34,55,\cdots$ است ، اولین و دومین جملات آن ۱ و ۱ هستند ، و جملات بعد ، مجموع دو جمله های قبل است. اعداد در این مجموعه ، **اعداد فیبو نا چی Fibonacci Numbers** نامیده می شوند. یک الگوریتم بنویسید که ورودی آن یک عدد صحیح مثبت n است و خروجی آن n تای اول اعداد فیبو نا چی است.

۵ - یک الگوریتم بنویسید که ورودی آن دو عدد صحیح n و k است و خروجی آن $\binom{n}{k}$ است.

۶ - یک الگوریتم بنویسید که ورودی آن یک لیست از اعداد $(x_1,x_2\cdots,x_n)$ است و خروجی آن کلمه yes است ، اگر لیست دارای ورودی های مکرر باشد ، در غیر این صورت خروجی کلمه no است.

۷ - همان طور که در مورد الگوریتم شماره ۵ بخش ۳.۱۰ گفتیم ، این الگوریتم کار نمی کند اگر طول لیست ۱ یا 0 باشد. آنرا تغییر دهید تا کار کند.

۸ - یک الگوریتم بنویسید که ورودی آن یک عدد صحیح n است ، و خروجی آن ردیف n ام مثلث پاسکال است.

پاسخ تمرینات ۱۰.۳ — ۱۰.۲ — ۱۰.۱

۱ - خروجی الگوریتم زیر را پیدا کنید.

```
x := 1
y := 10
while x^2 < y do
    y := y + x
    x := x + 1
end
output x
output y
```

پاسخ

در جدول زیر ، ستون دوم مقادیراولیه x و y است. ستون سوم مقادیر x و y در تکرار اول. ستون چهارم مقادیر x و y در تکرار دوم. ستون پنجم مقادیر x و y در تکرار سوم. پس مقادیر خروجی $x = ۴$ و $y = ۱۶$ است.

شماره تکرار		۱	۲	۳
x	۱	۲	۳	۴
y	۱۰	۱۱	۱۳	۱۶

۲ - خروجی الگوریتم زیر را پیدا کنید

```
a := 0
b := 3
for i := 1 to 8 do
    if a < b then
        a := a + i
    else
        b := b + a
    end
end
output a
output b
```

پاسخ

iteration (i)		1	2	3	4	5	6	7	8
a	0	1	3	3	7	7	13	13	21
b	3	3	3	6	6	13	13	26	26

جدول بالا ، مانند جدول تمرین ۱ است. ردیف اول شماره تکرار است ردیف دوم عدد 0 مقدار اولیه a است ردیف سوم عدد ۳ مقدار اولیه b است. بقیه مقادیر a و b هستند از تکرار اول تا هشتم. پس خروجی $a = ۲۱$ و $b = ۲۶$

۳ - طول یک لیست، یک عدد زوج است. اگر لیست $X=(3,5,8,4,6,8,7,4,2,3).$ باشد ، خروجی برای این لیست را پیدا کنید.

Algorithm

Input: $X=(x_1,x_2,\ldots,x_n)$
begin
  for $i:=1$ **to** $\frac{n}{2}$ **do**
    $k:=2i$
    $x_k:=x_k+1$
  end
  for $j:=1$ **to** $\frac{n}{2}$ **do**
    $k:=2j-1$
    $x_k:=x_k-1$
  end
  output X
end

پاسخ

اولین $for\ loop$ عدد ۱ را به هر یک از ورودی های لیست که دارای اندیس زوج هستند اضافه می کند. به عبارت دیگر ، ۱ به ورودی های $x_۲, x_۴, x_۶, x_۸, x_{۱۰}$

دومین $for\ loop$ عدد ۱ را از هر ورودی لیست که دارای اندیس فرد است کم می کند. به عبارت دیگر ۱ را از ورودی های $x_۱, x_۳, x_۵, x_۷, x_۹$ کسر می کند.

پس خروجی $X = (۲, ۶, ۷, ۵, ۵, ۹, ۶, ۵, ۱, ۴)$ است.

۴ - **دنباله فیبو نا چی Fibonacci Sequence** دنباله ۱, ۱, ۲, ۳, ۵, ۸, ۱۳, ۲۱, ۳۴, ۵۵, ... است ، اولین و دومین جملات آن ۱ و ۱ هستند ، و جملات بعد ، مجموع دو جمله های قبل است. اعداد در این مجموعه ، **اعداد فیبو نا چی Fibonacci Numbers** نامیده می شوند. یک الگوریتم بنویسید که ورودی آن یک عدد صحیح مثبت n است و خروجی آن n تای اول اعداد فیبو نا چی است.

پاسخ

Algorithm: to compute the first n Fibonacci numbers

Input: An integer n for which $n \geq 2$

Output: The first n Fibonacci numbers

begin

- $x := 1$ ..... x is the 1st Fibonacci number
- $y := 1$ ..... y is the 2nd Fibonacci number
- **output** x ..... output 1st Fibonacci number
- **output** y ..... output 2nd Fibonacci number
- $i := 2$ ..... i is # of Fibonacci numbers outputted so far
- **while** $i < n$ **do**
  - $z := x + y$ ..... z is most recent Fibonacci number
  - **output** z ..... output most recent Fibonacci number
  - $i := i + 1$ ..... update i
  - $x := y$ ..... x now second-most-recent Fibonacci number
  - $y := z$ ..... y now most recent Fibonacci number
- **end**

end

۵ ـ یک الگوریتم بنویسید که ورودی آن دو عدد صحیح n و k است و خروجی آن $\binom{n}{k}$ است.

پاسخ

یاد آوری :اگر $n \leq 0$, اما $k < 0$ یا $k > n$ باشد ، پس $\binom{n}{k} = 0$ است.همچنین اگر $k = 0$ یا $k = n$ باشد ، پس $\binom{n}{k} = 0$ است. در غیر این صورت داریم.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+3)(n-k+2)(n-k+1)}{k!}$$
$$= \frac{(n-k+1)(n-k+2)(n-k+3)\cdots(n-2)(n-1)\,n}{1\cdot 2\cdot 3\cdots(k-2)(k-1)\,k}.$$

الگوریتم ما عملیات بالا را انجام می دهد. ابتدا $y := ۱$ قرار می دهد ، سپس یک $for\ loop$ بکار می برد تا y را در $(n-k+۱)$ ضرب کند ، سپس در $(n-k+۲)$, سپس در $(n-k+۳)$ و الی آخر ضرب کند. این کار را ادامه می دهد تا به ضرب در n برسد. سپس یک $for\ loop$ بکار می برد تا حاصلضرب بدست آمده بر ۱ , سپس بر ۲ , سپس بر ۳ , و الی آخر، بر k تقسیم کند.

Algorithm: computes $\binom{n}{k}$

```
Input: Integers n and k, with n ≥ 0
Output: (n k)
begin
    if (n ≤ 0) ∨ ((k < 0) ∨ (k > n)) then
        output 0 ..... in this case (n k) = 0
    else
        if (k = 0) ∨ (k = n) then
            output 1 ..... in this case (n k) = 1
        else
            y := 1 ..... y is initially 1
            for i := n − k + 1 to n do
                y := y · i ..... multiply y by i, for each n − k + 1 ≤ i ≤ n
            end
            for i := 1 to k do
                y := y / i ..... divide y by i, for each 1 ≤ i ≤ k
            end
            output y ..... now y = (n k)
        end
    end
end
```

۶ - یک الگوریتم بنویسید که ورودی آن یک لیست از اعداد $(x_1, x_2 \cdots, x_n)$ است و خروجی آن کلمه yes است ، اگر لیست دارای ورودی های مکرر باشد ، در غیر این صورت خروجی کلمه no است.

پاسخ

Algorithm

Input: A list of numbers $x_1, x_2, x_3 \ldots, x_n$
Output: "YES" if the list has repetition, otherwise "NO"
begin
  $match :=$ NO
  for $i := 1$ **to** $n-1$ **do**
    for $k = i+1$ **to** n **do**
      if $x_i = x_k$ **then**
        $match :=$ YES
      end
    end
  end
end
output $match$

۷ - همان طور که در مورد الگوریتم شماره ۵ بخش ۳.۵ گفتیم ، این الگوریتم کار نمی کند اگر طول لیست ۱ یا 0 باشد. آنرا تغییر دهید تا کار کند.

پاسخ

(Better Bubble Sort) sorts any list

Input: A list $X = (x_1, x_2, \ldots, x_n)$ of numbers
Output: The list sorted into numeric order
begin
  if $0 \le n \le 1$ **then**
    output X ..... X is already sorted
  else
    for $k := 1$ **to** $n-1$ **do**
      for $i := 1$ **to** $n-k$ **do**
        if $x_i > x_{i+1}$ **then**
          $temp := x_i$ ..... temporarily holds value of x_i
          $x_i := x_{i+1}$
          $x_{i+1} := temp$ ..... now x_i and x_{i+1} are swapped
        end
      end
    end
    output X ..... now X is sorted
  end
end

۸ - یک الگوریتم بنویسید که ورودی آن یک عدد صحیح n است ، و خروجی آن ردیف n ام مثلث پاسکال است.

پاسخ

برای ورودی n, خروجی $n+1$ دنباله زیر است.

$$\binom{n}{0},\ \binom{n}{1},\ \binom{n}{2},\ \binom{n}{3},\ \dots,\ \binom{n}{n}$$

ابتدا یک $for\ loop$ لازم داریم تا کار زیر را انجام دهد.

for $k := 0$ **to** n **do**
  $y := \binom{n}{k}$
  output y
end

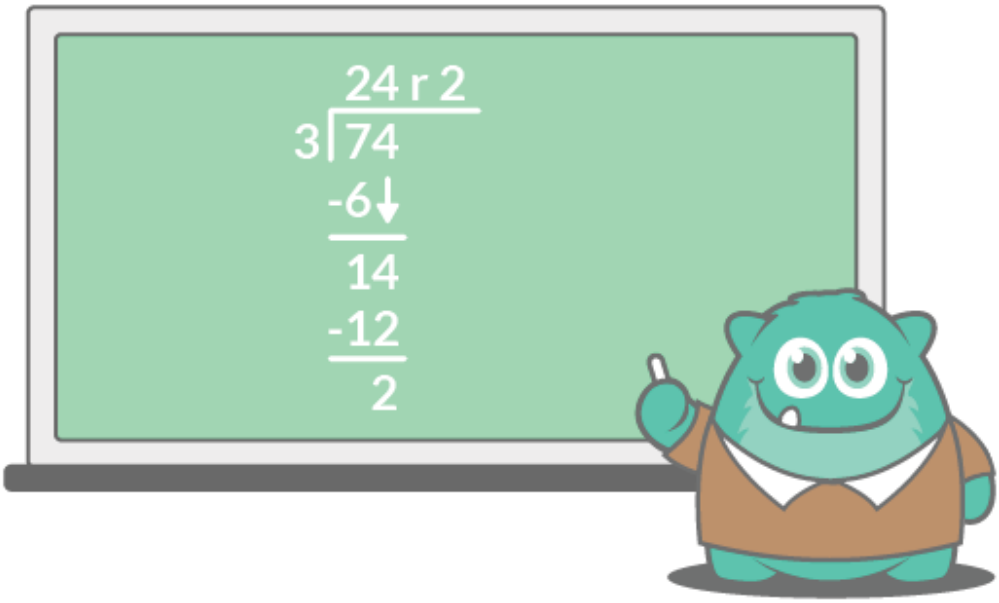
برای اتمام کار ، باید خطوطی که $y := \binom{n}{k}$ انجام می دهد اضافه کنیم. از تمرین شماره ۵ می توان استفاده کرد.

Algorithm: computes the nth row of Pascal's triangle

```
Input: Integer n with n ≥ 0
Output: nth row of Pascal's triangle
begin
    for k := 0 to n do
        if (k = 0) ∨ (k = n) then
            output 1 ..... in this case (n k) = 1
        else
            y := 1 ..... y is initially 1
            for i := n − k + 1 to n do
                y := y · i
            end
            for i := 1 to k do
                y := y / i
            end
            output y ..... now y = (n k)
        end
    end
end
```

۱۰.۴ - الگوریتم تقسیم Division Algorithm

در حساب دوره ابتدایی دیده ایم که اگر مثلا عدد ۷۴ را بر ۳ تقسیم کنیم ، خارج قسمت ۲۴ و باقیمانده ۲ است. عدد ۷۴ را **بخشی یا مقسوم Dividend** عدد ۳ را **بخشیاب یا مقسوم علیه Divisor** عدد ۲۴ را **خارج قسمت Quotient** و عدد ۲ را **باقیمانده Remainder** می نامند.



پس اگر یک عدد صحیح a بر یک عدد صحیح $b > 0$ تقسیم کنیم ، نتیجه یک خارج قسمت q و باقیمانده r است ، بطوری که $0 \leq r < b$ باشد ، پس می توان نوشت.

$$a = qb + r \quad \text{و} \quad 0 \leq r < b$$

حالا می خواهیم یک الگوریتم بنویسیم که ورودی آن دو عدد صحیح $a \geq 0$ و $b > 0$ و خروجی آن دو عدد q و r است ، بطوری که $a = qb + r$ و $0 \leq r < b$ باشد. یعنی ، خروجی خارج قسمت و باقیمانده نتیجه تقسیم a بر b است.

برای این که ببینیم چگونه پیش برویم ، ملاحظه می کنید اگر $a = qb + r$ باشد ، پس

$$a = \underbrace{b + b + b + \cdots + b}_{q \text{ times}} + r,$$

است. در عبارت بالا $q\ times$ یعنی q مرتبه. در صورتی که $r < b$ باشد. این یعنی اگر بطور مکرر b را از a کم کنیم تا یک عدد نا منفی r بدست آوریم که $r < b$ باشد. پس q تعداد دفعاتی است که b را از a کم کرده ایم. الگوریتم ما درست این کارا را انجام می دهد. الگوریتم عمل کم

کردن b از a را آنقدر ادامه می دهد تا یک پاسخ کوچک تر از b بدست آورد. متغیر q فقط تعداد دفعاتی که b ها کسر شده اند می شمرد.

Algorithm 6: The division algorithm

Input: Integers $a \geq 0$ and $b > 0$
Output: Integers q and r for which $a = qb + r$ and $0 \leq r < b$
begin
  $q := 0$ ..... so far we have subtracted b from a zero times
  while $a \geq b$ **do**
    $a := a - b$ .....subtract b from a until $a \geq b$ is no longer true
    $q := q + 1$ ..... q increases by 1 each time a b is subtracted
  end
  $r := a$ .....a now equals its original value, minus q b's
  output q
  output r
end

الگوریتم تقسیم به مصر قدیم و بابلی ها می‌رسد. آلبته آن زمان کامپیوتر وجود نداشت تا آنرا در کامپیوتر اجر کنند.
این الگوریتم بخاطر مفید بودن و مهم بودن آن ، تا بحال باقی مانده است.

امر مسلم ۱۰.۴.۱
الگوریتم تقسیم اگر دو عدد صحیح a و $b > 0$ داشته باشیم ، پس دو عدد q و r وجود دارند ، بطوری که $a = bq + r$ و $0 \leq r < b$

امر مسلم بالا برای اثبات بسیاری از قضایای مربوط به اعداد و ساختمان های ریاضی و سیستم ها کار برد دارد. توجه دارید که امر مسلم بالا نمی گوید لازم است که $a \geq 0$ باشد. اما الگوریتم شماره ۶ می گوید $a \geq 0$ باشد. در حقیقت الگوریتم تقسیم برای هر مقدار a کار می کند. مثلا

$$-۱۷ = -۶ * ۳ + ۱$$

۱۰.۵ - پروازه ها Procedures

توضیح : پردازه یک برنامه کوچکتر است که بخشی از یک برنامه اصلی است.

هنگام نوشتن یک الگوریتم ، ممکن است مجبور شویم گروهی از دستور العمل ها را چندین مرتبه بکار بریم. مثلاتصور کنید یک الگوریتم باید یک یا چند لیست را مرتب کند. هر مرتبه ، باید یک دستور العمل برای مرتب کردن حبابی ، در آن الگوریتم بگنجانیم. نوشتن مجدد دستور العمل ، خسته کننده است و بازده کم دارد.
برای رفع این مساله ، بیشتر زبان های برنامه نویسی ، ایجاد پردازه ها را مجاز کرده اند. این پردازه ها یک الگوریتم کوچک است که یک کار مشخص انجام می دهد. بطور کلی ، یک پردازه ، مثل یک تابع $f(x)$ یا $f(x,y)$ است که ما یک مقدار به تابع می دهیم و یک نتیجه بدست می آوریم.
در ذیل یک الگوریتم است که $n!$ را محاسبه می کند.

```
Procedure Fac(n)
begin
    if n = 0 then
        return 1 ..... because 0! = 1
    else
        y := 1
        for i := 1 to n do
            y := y · i
        end
        return y ..... now y = n!
    end
end
```

حالا این پردازه مثل یک تابع بنام Fac عمل می کند. این پردازه یک عدد n به عنوان ورودی می گیرد ، مقدار $y = n!$ را بر می گرداند. این موضوع در دستور کلمه **return** به معنی **به خروجی بفرست** ، مشخص شده است. مثلا $Fac(۵) = ۱۲۵$ ، $Fac(۴) = ۲۴$ ، $Fac(۳) = ۶$
حالا که پردازه را تعریف کردیم ، آنرا در یک الگوریتم که $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ را محاسبه می کند ، بکار می بریم.

Algorithm 7: to compute $\binom{n}{k}$

Input: Integers n and k, with $n \geq 0$
Output: $\binom{n}{k}$
begin
  if $(k<0) \vee (k>n)$ **then**
    output 0 ..... in this case $\binom{n}{k}=0$
  else
    output $\frac{\mathrm{Fac}(n)}{\mathrm{Fac}(k)\cdot \mathrm{Fac}(n-k)}$ .....procedure `Fac` is *called* here
  end
end

اگر یک الگوریتم ، مثل بالا ، یک پردازه قبلا تعریف شده را بکار می برد ، می گوییم الگوریتم پردازه را **فرا می خواند calls**
مثال بعدی برای یک پردازه ، بزرگ ترین Largest نامیده می شود. ورودی آن یک لیست $(x_1, x_2, \cdots, x_n)$ از اعداد است. و بزرگ ترین ورودی را به خروجی می فرستد. مثلا $Largest(7, 2, 3, 8, 4) = 8$ است. این پردازه باز نویسی الگوریتم ۴ است که به صورت یک پردازه نوشته شده است.

Procedure Largest($x_1, x_2, x_3, \ldots, x_n$)

begin
  $biggest := x_1$ .....this is the largest value found so far
  for $i := 1$ **to** n **do**
    if $biggest < x_i$ **then**
      $biggest := x_i$ .....this is the largest value found so far
    end
  end
  return $biggest$
end

توجه :می دانید که $4! = 1 * 2 * 3 * 4 = 24$ حال می توانیم بنویسیم $4! = 4 * 3 * 2 * 1 =$
این عمل را **باز گشت recursive** می نامند. پس اگر بجای

$$n! = 1 * 2 * 3 * \cdots * (n-2) * (n-1) * n$$

بنویسیم

$$n! = n * (n-1) * (n-2) * \cdots * 3 * 2 * 1$$

به صورت باز گشتی یا معکوس نوشته ایم.

تمرینات ۱۰.۵ – ۱۰.۴

۱ - یک پردازه بنویسید که ورودی آن یک لیست $X = (x_1, x_2, \cdots, x_n)$ از اعداد است و خروجی آن به ترتیب وارون است.

۲ - یک پردازه بنویسید که ورودی آن یک لیست $X = (x_1, x_2, \cdots x_n)$ از اعداد است و خروجی yes است ، اگر X به ترتیب عددی است ، و در غیر این صورت خروجی no است.

۳ - یک پردازه بنویسید که ورودی آن یک لیست 0 ها و 1 ها است به طول n یعنی $X = (0,0,1,0,1, \cdots, 1)$ است و خروجی تعداد 1 ها در X

۴ - یک پردازه بنویسید که ورودی آن یک لیست $X = (x_1, x_{2,\ldots}, x_n)$ از اعداد است و خروجی حاصل ضرب $x_1, x_2, \cdots x_n$ است.

۵ - یک پردازه بنویسید که ورودی آن لیست اعداد $X = (x_1, x_2, \cdots, x_n)$ و $Y = (y_1, y_2 \cdots, y_n)$ است و خروجی لیست $Z = (x_1, x_2, x_3, \cdots, x_n, y_n, \cdots, y_3, y_2, y_1)$ است.

۶ - یک پردازه بنویسید که ورودی آن لیست اعداد $X = (x_1, x_2, \cdots, x_n)$ و $Y = (y_1, y_2 \cdots, y_n)$ است و خروجی لیست $Z = (x_1 + y_1, x_2 + y_2, +x_3 + y_3, \cdots, x_n + y_n)$ است

پاسخ تمرینات ۱۰.۵ – ۱۰.۴

۱ - یک پردازه بنویسید که ورودی آن یک لیست $X = (x_1, x_2, \cdots, x_n)$ از اعداد است و خروجی آن به ترتیب وارون است.

پاسخ

```
Procedure Reverse(X)
begin
    Y := X ..... Y = (y1,...,yn) is a copy of X = (x1,...,xn)
    for i := 1 to n do
        yi := x_{n-i+1} ..... fill in Y as the reverse of X
    end
    return Y
end
```

۲ - یک پردازه بنویسید که ورودی آن یک لیست $X = (x_1, x_2, \cdots x_n)$ از اعداد است و خروجی yes است ، اگر X به ترتیب عددی است ، و در غیر این صورت خروجی no است.

پاسخ

```
Procedure Check(X)
begin
    ordered := YES ..... list assumed ordered until found not to be ordered
    i := 1
    while (ordered = YES) ∧ (i < n) do
        if x_i > x_{i+1} then
            ordered := NO
        end
        i := i + 1
    end
    return ordered
end
```

۳ - یک پردازه بنویسید که ورودی آن یک لیست 0 ها و 1 ها است به طول n یعنی $X = (0,0,1,0,1,\cdots,1)$ است و خروجی تعداد 1 ها در X

پاسخ

```
Procedure Ones(X)
begin
    total := 0 ..... so far total number of 1's found is zero
    for i := 1 to n do
        if x_i = 1 then
            total := total + 1
        end
    end
    return total
end
```

۴ - یک پردازه بنویسید که ورودی آن یک لیست $X = (x_1, x_{2,...}, x_n)$ از اعداد است و خروجی حاصل ضرب $x_1, x_2, \cdots x_n$ است.

پاسخ

```
Procedure Prod(X)
begin
    product := 1 for i := 1 to n do
        product := product · x_i
    end
    return prod
end
```

۵ - یک پردازه بنویسید که ورودی آن لیست اعداد $X = (x_1, x_2, \cdots, x_n)$ و $Y = (y_1, y_2 \cdots, y_n)$ است و خروجی لیست $Z = (x_1, x_2, x_3, \cdots, x_n, y_n, \cdots, y_3, y_2, y_1)$ است.

پاسخ

```
Procedure Glue(X,Y)
begin
    Z := (0,0,0,...,0) ..... a list of length 2n
    for i := 1 to n do
        z_i := x_i
        z_{n+1-i} := y_i
    end
    return Z
end
```

۶ - یک پردازه بنویسید که ورودی آن لیست اعداد $X = (x_1, x_2, \cdots, x_n)$ و $Y = (y_1, y_2 \cdots, y_n)$ است و خروجی لیست $Z = (x_1 + y_1, x_2 + y_2, +x_3 + y_3, \cdots, x_n + y_n)$ است

پاسخ

```
Procedure Add(X,Y)
begin
    Z := X ..... Z is a copy of X
    for i := 1 to n do
        z_i := z_i + y_i
    end
    return Z
end
```

۱۰.۶ ـ شمارش مراحل در الگوریتم ها Counting Steps in Algorithms

دانشمندان کامپیوتر مراقب کارایی الگوریتم هستند. یک الگوریتم باید کارش را در کوتاه ترین زمان ، با کمترین تعداد مراحل ، انجام دهد. مسلم است که تعداد مراحل لازم ، احتمالا بستگی به ورودی دارد. این سوال مهمی است.

یک الگوریتم X چند مرحله باید طی کند تا خروجی Y را پردازش کند؟

برای شروع ، فرض کنید یک الگوریتم دارای دستورالعمل های زیر است. به n هم یک عدد صحیح در خطوط قبل داده شده است.

```
for i := 1 to 3n do
    Command 1
    Command 2
end
Command 3
for j := 1 to n do
    for k := 1 to n do
        Command 4
    end
end
```

روی هم رفته چند دستور انجام می شوند؟ اولین $for\ loop$ باید $3n$ تکرار اجرا کند و در هر تکرار ۲ دستور ، پس می شود $3n * 2 = 6n$ سپس یک دستور یعنی $command$ ۳ اجرا می شود. سپس یک $for\ loop$ تو در تو ، در این حلقه تو در تو ، برای هر زوج "(i,k)، $1 \le j,k \le n$" دستور شماره ۴ یک مرتبه اجرا می شود. بر اساس اصل ضرب، $n * n = n^2$ چنین زوجی وجود دارد . پس دستور شماره ۴ باید n^2 مرتبه اجرا شود. لذا روی هم رفته $6n + 1 + n^2$ دستور باید اجرا شود.

حالا ، اجازه دهید مراحل دستورالعمل زیر را شمارش کنیم.

```
for i := 0 to n do
    for j := 0 to i do
        for k := 0 to j do
            Command 1
        end
    end
end
```

دستور ۱ برای هر یک از ترکیب های i,j,k با $0 \le k \le j \le i \le n$ اجرا می شود. هر یک از ترکیب ها مربوط است به یک لیست n ستاره و ۳ خط عمودی مانند $***|**|*|**\cdots*$

عدد k تعداد ستاره های سمت چپ اولین خط عمودی ، عدد j تعداد ستاره های سمت چپ دومین خط عمودی و عدد i تعداد ستاره های سمت چپ سومین خط عمودی است. چنین لیستی دارای طول $n+۳$ است. برای ساختن این لیست ، ۳ تا از $n+۳$ محل ها برای خطوط و بقیه را با ستاره پر می کنیم. تعداد $\binom{n+۳}{۳}$ از چنین لیستی وجود دارد. پس تعداد دفعاتی که دستور شماره ۱ اجرا می شود مطابق زیر است.

$$\binom{n+3}{3} = \frac{n(n-1)(n-2)}{3!} = \frac{n^3-3n^2+2n}{6} = \frac{1}{6}n^3 - \frac{1}{2}n^2 + \frac{1}{3}n.$$

در خاتمه این فصل ، دو الگوریتم متفاوت که هر دو یک کار انجام می دهند ، مقایسه می کنیم. این الگوریتم ها، در یک لیست شامل اعداد مرتب جستجو می کنند ، و اگر یک عدد مخصوص پیدا شود ، آن عدد را مشخص می کند. خواهیم دید الگوریتم دوم ، گر چه تقریبا پیچیده ، ولی کار آمد تر است.

هر یک از این الگوریتم ها یک عدد z و یک لیست اعداد مرتب $X = (x_۱, x_۲, \cdots, x_n)$ به عنوان ورودی دریافت می کند. لیست به این صورت $x_۱ \le x_۲ \le \cdots \le x_n$ مرتب است. خروجی کلمه Yes است اگر z مساوی یکی از اعداد درون لیست باشد ، در غیر این صورت خروجی کلمه No است.

اولین الگوریتم ، بنام **جستجوی ترتیبی Sequential Search** در لیست از چپ به راست سیر می کند ، اگر $z = x_k$ پیدا کرد متوقف می شود ، اگر پیدا نکرد تا انتهای لیست بدون پیدا کردن x_k به حرکت ادامه می دهد. یک متغیر $found$ یا مساوی Yes است یا مساوی No. الگوریتم با اختصاص دادن $found := No$ شروع می کند ، و آنرا تغییر می دهد اکر یک k پیدا کند بطوری که $z = x_k$ باشد. الگوریتم یک $while\ loop$ دارد که به حرکت ادامه می دهد تا زمانی که $found := No$ است و $k \le n$

Algorithm 8: sequential search

Input: A number z and a sorted list $X = (x_1, x_2, \ldots, x_n)$ of numbers
Output: "YES" if z appears in X; otherwise "NO"
begin
  $found :=$ NO ..... means z not yet found in X
  $k := 0$ ..... k is subscript for list entries x_k
  while $(found =$ NO$) \wedge (k < n)$ **do**
    $k := k + 1$ ..... go to next list entry
    if $z = x_k$ **then**
      $found :=$ YES ..... the number z appears in X
    end
  end
  output $found$
end

الگوریتم شماره ۸ دو دستور قبل از $while\ loop$ دارد. سپس $while\ loop$ حد اکثر n تکرار انجام می دهد. هر تکرار شامل دو دستور است. پس الگوریتم یک لیست بطول n را حد اکثر در $2n + 2$ مرتبه جستجو می کند. اگر z اصلا پیدا نشود و یا درست در آخرین مرحله پیدا شود ، $2n + 2$ مرتبه تکرار ، بد ترین حالت است. از طرف دیگر ، اگر $x_1 = z$ باشد ، پس االگوریتم بعد از ۴ مرحله ، متوقف می شود.

حالا ، یک الگوریتم دیگر طرح می کنیم که جستجوی لیست را به طریق دیگری انجام دهد. بر خلاف جستجوی ترتیبی که تمام ورودی های لیست را امتحان می کند ، این الگوریتم جدید ، تقریبا تمام ورودی ها را نا دیده می‌گیرد ، ولی پاسخ صحیح را بر می گرداند.

برای توضیح این ایده ، فرض کنید می خواهیم ببینیم آیا $z = 4$ در لیست زیر است یا نه.

$$X = (0,1,1,2,3,3,3,3,4,5,5,5,5,8,8,9)$$

اگر z در لیست باشد ، باید بین منتها آلیه سمت چپ یعنی $L = 1$ و منتها آلیه سمت راست یعنی $R = 16$ باشد.
توجه : در جدول پایین L حرف اول کلمه $Left$ است به معنی چپ ، M حرف اول $Middle$ است به معنی میان و R حرف اول $Right$ است به معنی راست.

$X =$	0	1	1	2	3	3	3	3	4	5	5	5	5	8	8	9
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
	↑							↑								↑
	$L = 1$							$M = \lfloor \frac{L+R}{2} \rfloor = 8$								$R = 16$

پس الگوریتم به وسط لیست می جهد یعنی $M = \lfloor \frac{L+R}{2} \rfloor = 8$، میانگین L و R گرد شده به یک عدد صحیح. عدد $z = 4$ که در جستجویش هستیم بزرگ تر از $x_M = 3$ است ، پس $z = 4$ سمت راست x_8 است. یعنی در قسمت سایه در جدول زیر.

$X =$	0	1	1	2	3	3	3	3	4	5	5	5	5	8	8	9
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
									↑			↑				↑
									$L=9$		$M = \lfloor \frac{L+R}{2} \rfloor = 12$					$R=16$

پس تجدید نظر می کنیم $L := M + 1$ و لذا وسط جدید می شود $M := \lfloor \frac{L+R}{2} \rfloor = 12$ ملاحظه می کنید که $x_M = x_{12} = 5$ است. و عدد $z = 4$ که در جستجویش هستیم کوچک تر از x_M است. پس z

در قسمت سایه دار جدول زیر است. لذا $R = M - ۱$ است و وسط جدید $M := \lfloor \frac{L+R}{2} \rfloor = 10.$ است.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
$X =$	0	1	1	2	3	3	3	3	4	5	5	5	5	8	8	9

$L=9$ $M=\lfloor \frac{L+R}{2} \rfloor =10$ $R=11$

باز هم $x_M = ۵$ است و عدد z که در جستجویش هستیم .کوچک تر از x_M است . پس آن در قسمت سایه دار جدول زیر است. $R := M - ۱$ را به روز می کنیم وسط جدید $M := \lfloor \frac{L+R}{2} \rfloor = 9.$ است.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
$X =$	0	1	1	2	3	3	3	3	4	5	5	5	5	8	8	9

$L=9$ $M=\lfloor \frac{L+R}{2} \rfloor =9$ $R=9$

حالا $L = R$ است. عدد دلخواه را پیدا کردیم و به این طریق بیشتر لیست را نا دیده گذ شتیم.

این لگاریتم جدید ، **جستجوی دو دویی Binary Search** نامیده می شود. کار این جستجو به این صورت است که پیوسته دو لیست مکان های چپ و راست ایجاد می کند و z را جستجو می کند که باید بین این دو لیست چپ و راست باشد. در هر تکرار ، یک وسط M محاسبه می شود.
اگر $x_M = z$ باشد ، z را پیدا کرده ایم. اگر $x_M < z$ باشد ، پس z سمت راست M است ، پس $M + ۱$ می شود L جدید ، اگر $x_M > z$ باشد ، پس z سمت چپ M است و لذا $M - ۱$ می شود R جدید. به این طریق L و R به هم نزدیک تر و نزدیک تر می شوند ، و در نتیجه z را به دام می اندازند، اگر واقعا X شامل z باشد.
اگر z در X نباشد ، پس در نهایت $L = R$ می شود. اینجا الگوریتم به پایان می رسد و گزارش می دهد که z در X نیست. **الگوریتم ۹ در صفحه بعد ملاحظه می کنید.**

اجازه دهید ، تعداد مراحل لازم برای انجام جستجوی دو دویی روی یک لیست بطول n را تجزیه و تحلیل کنیم. الگوریتم ۹ با ۳ دستور شروع می شود. یعنی مقادیر اولیه $found$ ، L, R را مشخص می کند. سپس به $while\ loop$ می رسد، تکرار می شود تا وقتی که $found = Yes$ یا $L = R$ بشود. این چند تکرار است؟ قبل از تکرار اول ، فاصله L و R مقدار $n - ۱$ است. در هر تکرار فاصله بین L و R حد اقل نصف است.

پس، بعد از اولین تکرار فاصله بین L و R کمتر از $\frac{n}{۲}$ است. بعد از تکرار دوم فاصله بین آنها کمتر از $\frac{n}{۲^۲} = \frac{n}{۲} * \frac{۱}{۲}$ است. بعد از تکرار سوم فاصله بین آنها کمتر از $\frac{n}{۲^۳} = \frac{n}{۲^۲} * \frac{۱}{۲}$ است. پس بعد از i تکرار ، فاصله آنها کمتر از $\frac{n}{۲^i}$ است. **بقیه تجزیه و تحلیل بعد از الگوریتم ۹**

Algorithm 9: binary search

Input: A number z, and a sorted list $X = (x_1, x_2, \ldots, x_n)$ of numbers
Output: "YES" if z appears in X; otherwise "NO"
begin
  $found := \text{NO}$ ..... this means z has not yet been found in X
  $L := 1$ ..... left end of search area is x_1
  $R := n$ ..... right end of search area is x_n
  while $(found = \text{NO}) \wedge (L < R)$ **do**
    $M := \left\lfloor \frac{L+R}{2} \right\rfloor$ ..... M is middle of search area
    if $z = x_M$ **then**
      $found := \text{YES}$ ..... the number z appears in X
    else
      if $z < x_M$ **then**
        $R := M - 1$ ..... if z is in X, it's between x_L and x_M
      else
        $L := M + 1$ ..... if z is in X, it's between x_M and x_R
      end
    end
  end
  output $found$
end

پس در بد ترین حالت ، $while\ loop$ برای اجرا ی دستور ها، i تکرار ، ادامه می دهد ، تا

$$\frac{n}{2^i} \le 1 < \frac{n}{2^{i-1}}$$

این کمترین i است که می توان مطمئن بود فاصله بین R و L کمتر از ۱ و در نتیجه صفر است.
طرفین رابطه بالا را در ۲^i ضرب می کنیم و نتیجه می شود

$$n \le ۲^i < ۲n$$

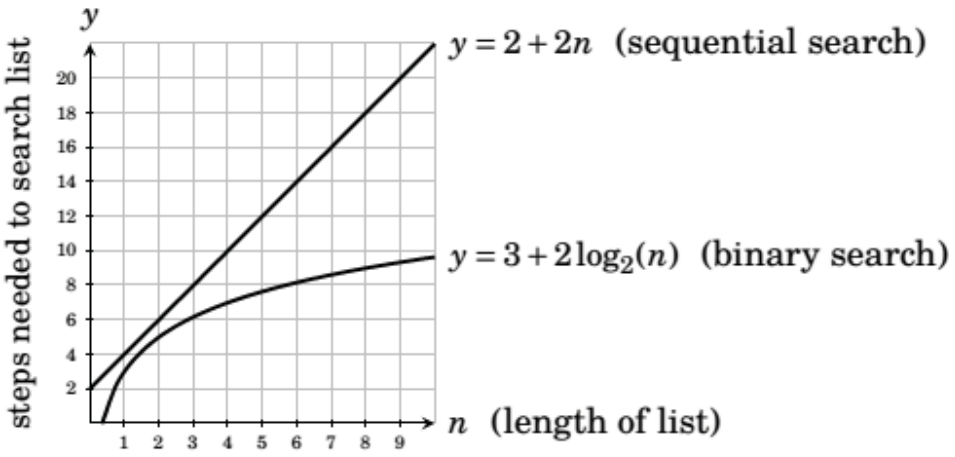
با لگاریتم گرفتن نتایج زیر را بدست می آوریم.

$$\begin{aligned} \log_2(n) &\le \log_2(2^i) < \log_2(2n) \\ \log_2(n) &\le i < \log_2(2) + \log_2(n) \\ \log_2(n) &\le i < 1 + \log_2(n). \end{aligned}$$

پس تعداد تکرار i, یک عدد صحیح است بین $\log_۲(n)$ و $۱ + \log_۲(n)$, یعنی
$i = \lceil \log_۲(n) \rceil$

خلاصه ، جستجوی دو دویی حد اکثر $۲\lceil \log_۲(n) \rceil + ۳$ مرحله طی می کند تا یک لیست به طول n را جستجو کند. بر عکس ، جستجوی ترتیبی $۲n + ۲$ مرحله طی می کند تا کار جستجو را تمام کند.

نمودار زیر تصویری است برای مقایسه جستجوی ترتیبی و جستجوی دو دویی
جستجوی ترتیبی :Sequential Search
جستجوی دو دویی :Binary Search



تمرینات ۱۰.۶

۱ - چند مرتبه دستور داخل $for\ loop$ اجر می شود؟

```
for i := 1 to 60 do
    for j := 1 to i do
        Command
    end
end
```

۲ - فرض کنید n یک عدد صحیح مثبت باشد. چند مرتبه یک دستور اجرا می شود؟ پاسخ بستگی به n دارد.

```
for i := 0 to n do
    for j := 0 to i do
        for k := 0 to j do
            for ℓ := 0 to k do
                Command
            end
        end
    end
end
```

۳ - چند مرتبه دستور اجرا می شود؟

```
for i := 1 to 2017 do
    if i is even then
        Command
    else
        Command
        Command
    end
end
```

۴ - اگر در الگوریتم جستجوی حبابی ، لیست ورودی $X = (x_1, x_2, \cdots, x_n)$ قبلا مرتب شده باشد ، چند مرحله انجام می دهد؟

Algorithm 5: (Bubble Sort) sorts a list

```
Input: A list X = (x1, x2, ..., xn) of numbers
Output: The list sorted into numeric order
begin
    for k := 1 to n − 1 do
        for i := 1 to n − k do
            if x_i > x_{i+1} then
                temp := x_i ..... temporarily holds value of x_i
                x_i := x_{i+1}
                x_{i+1} := temp .....now x_i and x_{i+1} are swapped
            end
        end
    end
    output X ..... now X is sorted
end
```

۵ - یک فرمول برای تعداد دفعاتی که الگوریتم زیر برای یک ورودی n اجرا می کند ، پیدا کنید.

Algorithm 2: computes the first n positive even integers

```
Input: A positive integer n
Output: The first n positive even integers
begin
    for i := 1 to n do
        y := 2 · i ..... y is the ith even integer
        output y
    end
end
```

۶ - یک فرمول برای تعداد دفعاتی که الگوریتم زیر اجرا می کنند هنگامی که ورودی ، یک لیست بطول n باشد.

Algorithm 4: finds the largest entry of a list

Input: A list $X = (x_1, x_2, \ldots, x_n)$
Output: The largest entry in the list
begin
  $biggest := x_1$ ..... this is the largest value found so far
  for $i := 1$ **to** n **do**
    if $biggest < x_i$ **then**
      $biggest := x_i$ ..... this is the largest value found so far
    end
  end
  output $biggest$
end

۷ - یک فرمول برای تعداد دفعاتی که الگوریتم تقسیم انجام می دهد هنگامی که ورودی دو عدد صحیح مثبت a و b باشد. پاسخ بستگی به a و b خواهد داشت.

Algorithm 6: The division algorithm

Input: Integers $a \geq 0$ and $b > 0$
Output: Integers q and r for which $a = qb + r$ and $0 \leq r < b$
begin
  $q := 0$ ..... so far we have subtracted b from a zero times
  while $a \geq b$ **do**
    $a := a - b$ .....subtract b from a until $a \geq b$ is no longer true
    $q := q + 1$ ..... q increases by 1 each time a b is subtracted
  end
  $r := a$ .....a now equals its original value, minus q b's
  output q
  output r
end

پاسخ تمرینات ۱۰.۶

۱ - چند مرتبه دستور داخل $for\ loop$ اجر می شود؟

```
for i := 1 to 60 do
    for j := 1 to i do
        Command
    end
end
```

پاسخ

دستور برای هر زوج (i,j) با $1 \le j \le i \le 60$ یک مرتبه اجرا می شود . چنین زوجی را می توان با یک لیست از ستاره و خط عمودی به صورت $* \cdots *** \mid * \cdots *** \mid * \cdots ***$ کد نویسی کرد با ۶۰ ستاره و دو خط عمودی. j تعداد ستاره ها ی قبل از اولین خط عمودی و i تعداد ستاره ها قبل از خط عمود دوم. توضیح: هر ستاره جانشین دستوری است که داخل حلقه دوم یعنی j است. این دستور می تواند هر چیزی باشد ، یک عدد ، یک حرف ، یک کلمه و یا حتی یک جمله باشد. هر خط عمود هم نشانه پایان یک حلقه کامل ، و چون دو حلقه تو در تو داریم ، پس فاصله این دو را با دو خط عمودی نشان می دهیم.

اگر $j \le 1$ باشد، اولین ورودی باید یک ستاره باشد. بقیه ورودی تشکیل یک لیست با ۵۹ ستاره و دو خط عمودی می دهد. طول هر لیست ۶۱ است. تعداد چنین لیستی $\binom{61}{2} = 1830$ است. پس تعداد دستور های صادر شده ۱۸۳۰ است.

۲ - فرض کنید n یک عدد صحیح مثبت باشد. چند مرتبه یک دستور اجرا می شود؟ پاسخ بستگی به n دارد.

```
for i := 0 to n do
    for j := 0 to i do
        for k := 0 to j do
            for ℓ := 0 to k do
                Command
            end
        end
    end
end
```

پاسخ

دستوربرای هر ترکیبی ازاعداد صحیح i,j,k,l با $0 \le l \le k \le j \le i \le n$ اجرا می شود. اینجا هم مانند تمرین شماره یک اجرای هر دستور را با یک ستاره و فاصله بین هر حلقه با یک خط عمود ، نشان می دهیم. چون چهار حلقه تو در تو داریم ، پس چهار خط عمودی تصور می کنیم و لیست هایی با n ستاره. اینجا هم l تعداد ستاره های سمت چپ خط اول ، k تعداد ستاره های سمت چپ خط دوم ،

j تعداد ستاره های سمت چپ خط سوم و i تعداد ستاره های سمت چپ خط چهارم. چنین لیستی دارای طول $n+۴$ است. پس تعداد لیست ها $\binom{n+۴}{۴}$ است. لذا $command$ یا دستور $\binom{n+۴}{۴}$ مرتبه اجرا می شود.

۳ - چند مرتبه دستور اجرا می شود؟

```
for i := 1 to 2017 do
    if i is even then
        Command
    else
        Command
        Command
    end
end
```

پاسخ

تعداد $\frac{۲۰۱۸}{۲} = ۱۰۰۹$ اعداد فرد بین ۱ و ۲۰۱۷ وجود دارد و تعداد ۱۰۰۸ اعداد زوج بین ۱ و ۲۰۱۷ وجود دارد. چون دستور برای هر عدد صحیح زوج یک مرتبه صادر می شود و برای هر عدد فرد دو مرتبه صادر می شو د , پس تعداد اجرای دستور ها $۱۰۰۹ * ۲ + ۱۰۰۸ = ۲۰۲۶$ است.

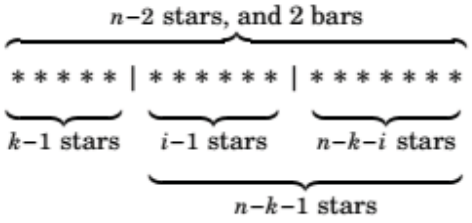
۴ - اگر در الگوریتم جستجوی حبابی ، لیست ورودی $X = (x_۱, x_۲, \cdots, x_n)$ قبلا مرتب شده باشد ، چند مرحله انجام می دهد؟

Algorithm 5: (Bubble Sort) sorts a list

```
Input: A list X = (x1,x2,...,xn) of numbers
Output: The list sorted into numeric order
begin
    for k := 1 to n − 1 do
        for i := 1 to n − k do
            if x_i > x_{i+1} then
                temp := x_i ..... temporarily holds value of x_i
                x_i := x_{i+1}
                x_{i+1} := temp .....now x_i and x_{i+1} are swapped
            end
        end
    end
    output X ..... now X is sorted
end
```

پاسخ در صفحه بعد

گزاره if داخل حلقه های تو در تو برای هر زوج اعداد صحیح (k,i) با $۱ \le k \le n-۱$ و $۱ \le i \le n-k$ یک مرتبه اجرا می شود. می توانیم چنین زوج هایی را به صورت لیست هایی به طول n تصور کنیم ، که از $n-۲$ ستاره و ۲ خط عمود درست شده باشند. تعداد $k-۱$ ستاره قبل از خط اول ، و $i-۱$ ستاره قبل از خط دوم.



مثلا ، فرض می کنیم $n=۸$ باشد ، پس ** | * | *** مربوط است به $(k,i)=(۴,۲)$ در صورتی که *** || *** مربوط است به $(k,i)=(۴,۱)$ همچنین |******| یعنی $(k,i)=(۱,۷)$

و ****** || یعنی $(k,i)=(۱,۱)$ ۰ تعداد چنین لیستی $\binom{n}{۲}=\frac{n(n-۱)}{۲}=\frac{۱}{۲}n^۲-\frac{۱}{۲}n$ است.

یعنی تعداد راه ها ی انتخاب ۲ از n جا برای خطوط. پس تعداد اجرای گزاره if عبارت است از $\frac{۱}{۲}n^۲-\frac{۱}{۲}n$ مرتبه. اما $x_i > x_{i+۱}$ همیشه غلط است ، لذا سه گزاره های داخل if اجرا نمی شوند.

پس الگوریتم $\frac{۱}{۲}n^۲-\frac{۱}{۲}n$ مرحله انجام می دهد.

۵ - یک فرمول برای تعداد دفعاتی که الگوریتم زیر برای یک ورودی n اجرا می کند ، پیدا کنید.

Algorithm 2: computes the first n positive even integers

```
Input: A positive integer n
Output: The first n positive even integers
begin
    for i := 1 to n do
        y := 2·i  ..... y is the ith even integer
        output y
    end
end
```

پاسخ

برای هر i بین ۱ و n الگوریتم دو مرحله اجرا می کند ، پس روی هم رفته $۲n$ اجرا می شود.

۶ - یک فرمول برای تعداد دفعاتی که الگوریتم زیر اجرا می کنند هنگامی که ورودی ، یک لیست بطول n باشد.

Algorithm 4: finds the largest entry of a list

Input: A list $X = (x_1, x_2, \ldots, x_n)$
Output: The largest entry in the list
begin
  $biggest := x_1$ ..... this is the largest value found so far
  for $i := 1$ **to** n **do**
    if $biggest < x_i$ **then**
      $biggest := x_i$ ..... this is the largest value found so far
    end
  end
  output $biggest$
end

پاسخ

الگوریتم با یک $(biggest := x_1)$ شروع می شود. سپس یک گزاره if را n مرتبه اجرا می کند پس روی هم $n + 1$ اجرا داریم.

۷ - یک فرمول برای تعداد دفعاتی که الگوریتم تقسیم انجام می دهد هنگامی که ورودی دو عدد صحیح مثبت a و b باشد. پاسخ بستگی به a و b خواهد داشت.

Algorithm 6: The division algorithm

Input: Integers $a \geq 0$ and $b > 0$
Output: Integers q and r for which $a = qb + r$ and $0 \leq r < b$
begin
  $q := 0$ ..... so far we have subtracted b from a zero times
  while $a \geq b$ **do**
    $a := a - b$ .....subtract b from a until $a \geq b$ is no longer true
    $q := q + 1$ ..... q increases by 1 each time a b is subtracted
  end
  $r := a$ .....a now equals its original value, minus q b's
  output q
  output r
end

پاسخ

چهار جمله خارج از $while\ loop$ وجود دارد. حلقه $\left\lfloor \frac{a}{b} \right\rfloor$ تکرار دارد. هر تکرار دو جمله اجرا می کنند. پس پاسخ $\left\lfloor \frac{a}{b} \right\rfloor$ ۲ + ۴ است.