



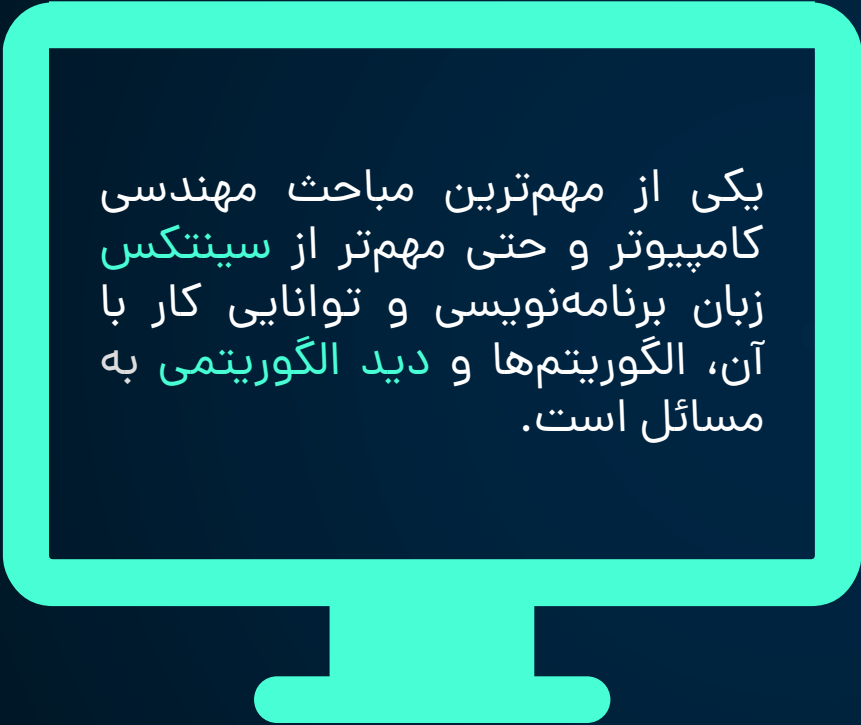
شبه کد

جلسه سوم

بسم الله الرحمن الرحيم



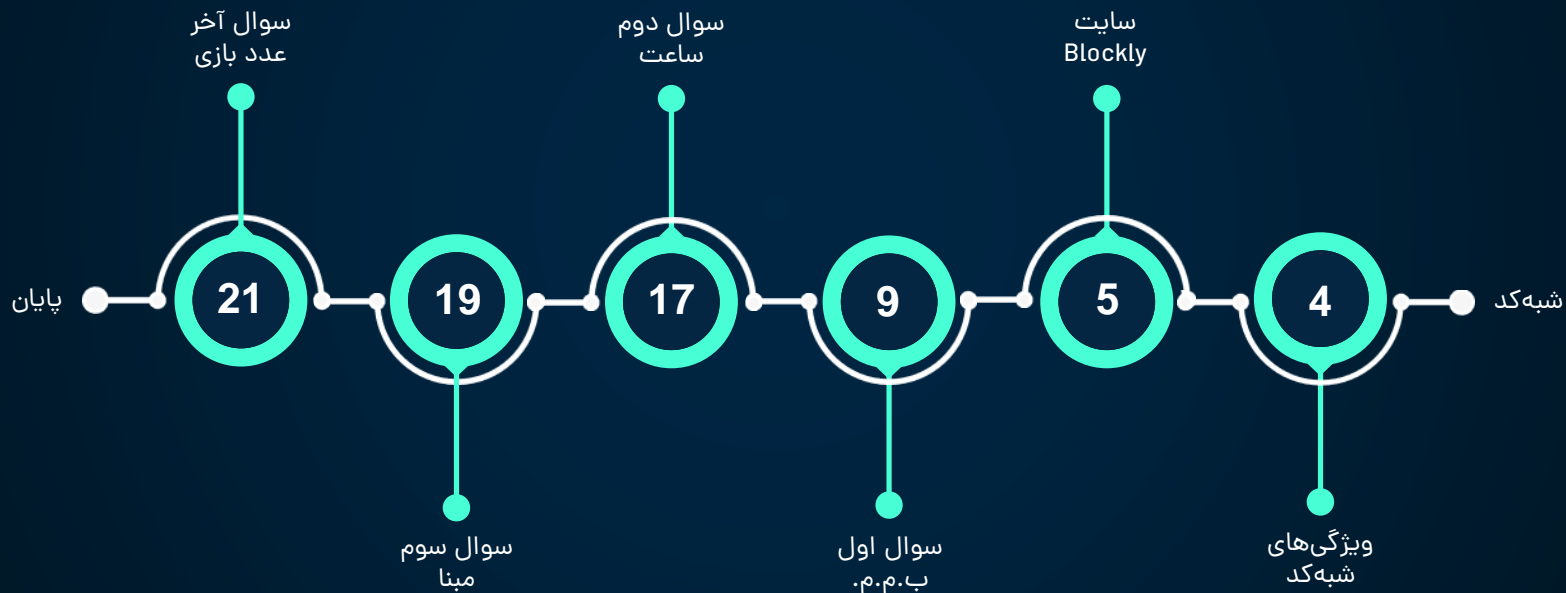
کارگاه مبانی برنامه نویسی - دانشکده مهندسی کامپیوتر دانشگاه هوایی شهید ستاری



یکی از مهمترین مباحث مهندسی  
کامپیوتر و حتی مهم‌تر از سینتکس  
زبان برنامه‌نویسی و توانایی کار با  
آن، الگوریتم‌ها و دید الگوریتمی به  
مسائل است.

برای تمرین این توانایی، در طی این  
جلسه بدون استفاده از زبان  
برنامه‌نویسی و با استفاده از شبه‌کدها  
سعی به حل سوالات با تکیه بر  
الگوریتم داریم.

# فهرست



# ویژگی‌های شبه کد

- ✓ ساده‌تر کردن حل مسئله با ایجاد دید بهتر نسبت به نیازهای سوال
- ✓ قابل فهم بودن برای انسان‌ها به دلیل نزدیکی آن به زبان انسان
- ✓ عدم نیاز به تسلط بر زبان‌های برنامه‌نویسی
- ✓ کمک به پیاده‌سازی راحت‌تر الگوریتم‌های مورد نیاز
- ✓ غیر قابل اجرا در کامپیوتر

# سایت Blockly

همان‌طور که گفته شد، شبه‌کدها به صورت نوشته‌های متنی نوشته می‌شوند که قابل اجرا نیستند. اما برای نزدیک‌تر شدن به مفهوم کد، ما از سایتی کمک می‌گیریم که شبه‌کد را به برنامه تبدیل می‌کند تا قابل اجرا باشد و خروجی آن مشاهده شود. (در ابتدا با فیلترشکن وارد شوید، سپس می‌توانید فیلترشکن خود را خاموش کنید)



[Blockly](#)

Scratch

Blockly

Hopscotch

# VPL

برای مطالعه

VPL که مخفف Visual Programming language به معنای زبان برنامه‌نویسی دیداری می‌باشد، به هر زبان برنامه‌نویسی‌ای گفته می‌شود که برای نوشتن یک برنامه به کمک آن، نیازمند ابزارها و دستورات گرافیکی هستیم.

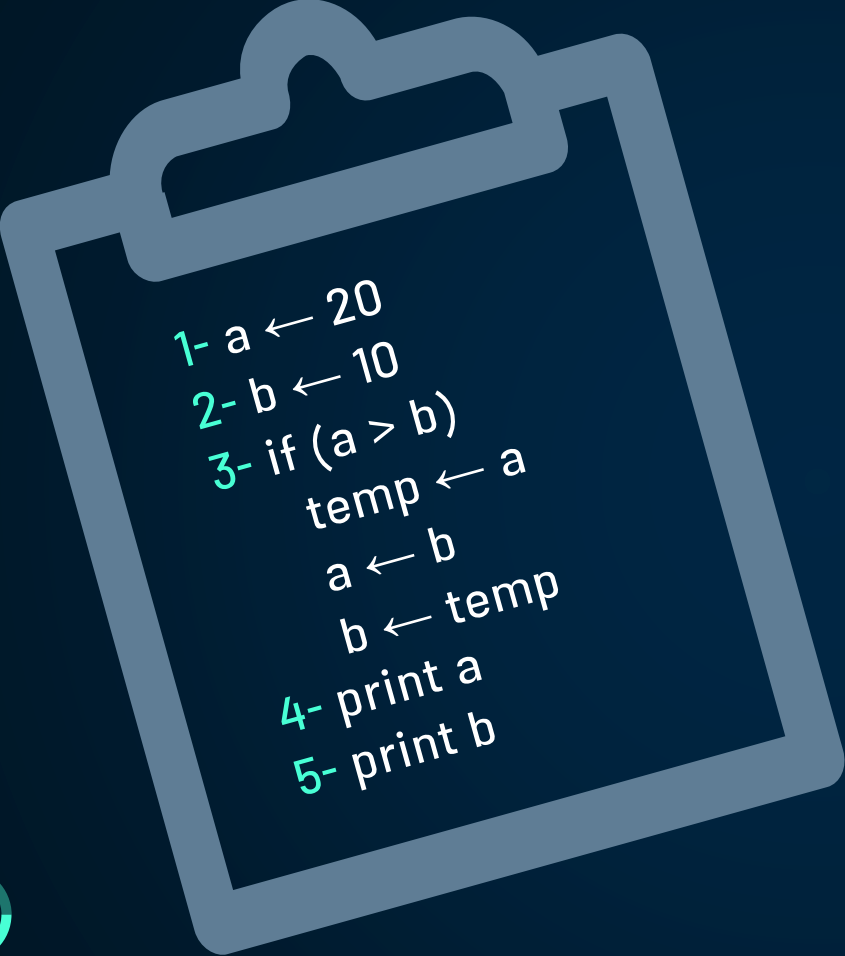
در این زبان‌ها، سر و کار برنامه‌نویس با بلاک‌ها، باکس‌ها و فلش‌هاست و یا هر شکل خاص دیگری که به فراخور آن محیط طراحی شده‌است.

زبان‌های VPL متنوعی مانند Scratch، Blockly، Bubble و ... وجود دارد.

Bubble

Scratch

Blockly



```
1- a ← 20
2- b ← 10
3- if (a > b)
    temp ← a
    a ← b
    b ← temp
4- print a
5- print b
```

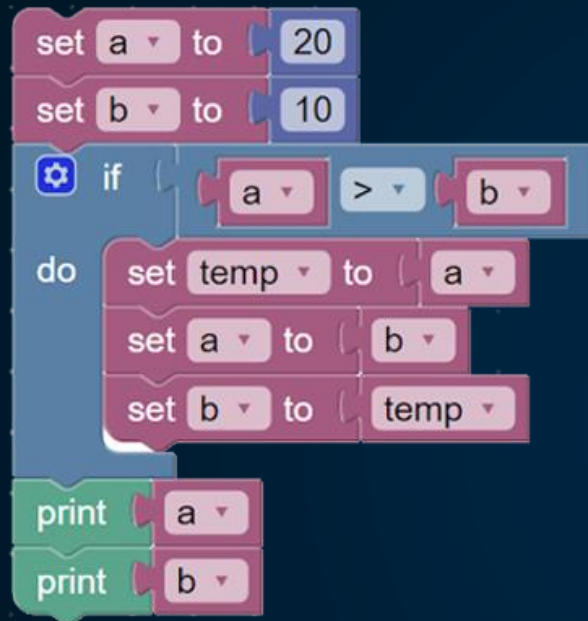
برای تمرین و آشنایی با سایت، سعی




کنید شبکه‌کد روبه‌رو را در سایت Blockly طراحی کنید.

شبکه‌کد جابه‌جا کردن دو عدد در صورت  
بزرگ‌تر بودن عدد اول:

خروجی حاصل باید چنین شکلی باشد:



حال با زدن روی دکمه‌ی play یعنی  در گوشه‌ی بالا سمت راست صفحه می‌توانید خروجی شبه‌کد خود را در دو مرحله مشاهده کنید.

blockly-demo.appspot.com says

20

OK

blockly-demo.appspot.com says


10


OK



# سوال اول: ب.م.م.

فرض کنید برای حل مسئله‌ای نیاز به پیدا کردن بزرگ‌ترین مقسوم علیه مشترک دو عدد دارید. 

شبه‌کدی بنویسید که بتواند این کار را انجام دهد و با داشتن ۲ عدد، ب.م.م. آن‌ها را حساب کند و خروجی دهد. 

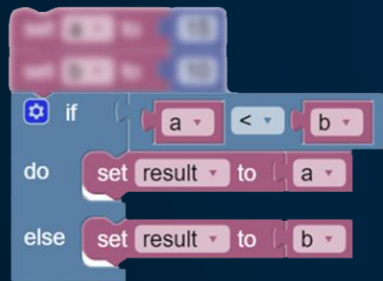
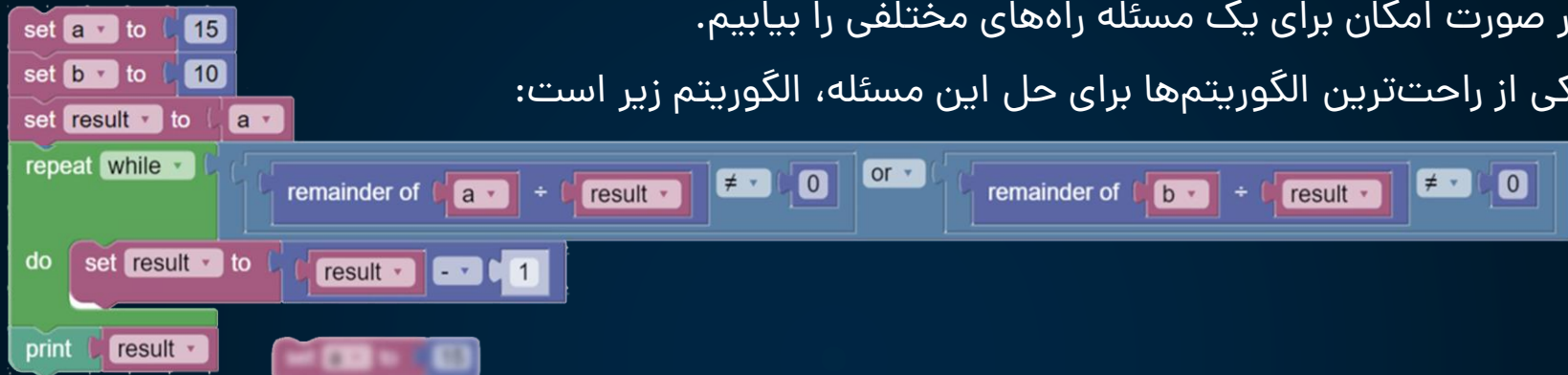
برای حل اکثر مسائل، بیش از یک راه‌حل وجود دارد و با تغییر طرز تفکر و در اصل تغییر الگوریتم مورد استفاده، می‌توان به طریق دیگری مسئله را مدل‌سازی و حل کرد؛ به شکلی که از نظر زمانی سریع‌تر به جواب برسیم یا برای رسیدن به جواب، حافظه‌ی کمتری اشغال کنیم. 

البته در درس مبانی کامپیوتر، هدف یافتن الگوریتم بهینه نیست؛ اما گاهی سعی می‌کنیم

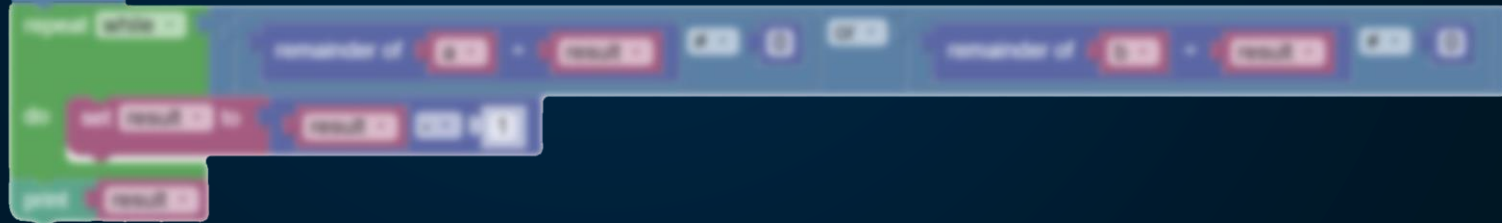


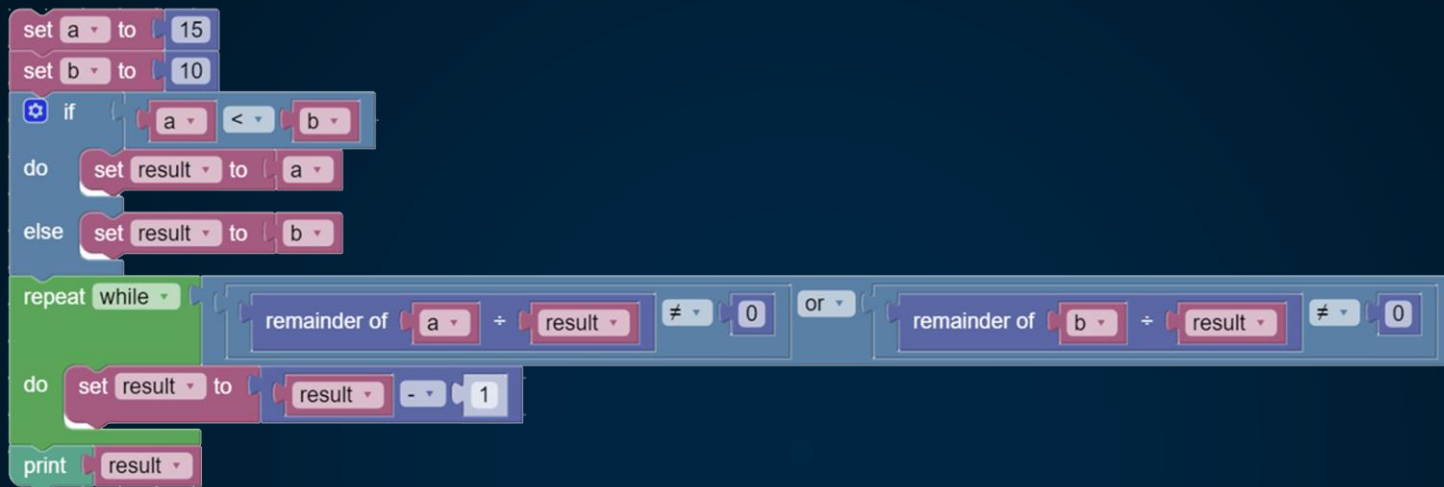
در صورت امکان برای یک مسئله راه‌های مختلفی را بیابیم.

یکی از راحت‌ترین الگوریتم‌ها برای حل این مسئله، الگوریتم زیر است:



به نظر شما این شرط چه تغییری در الگوریتم ایجاد می‌کند؟

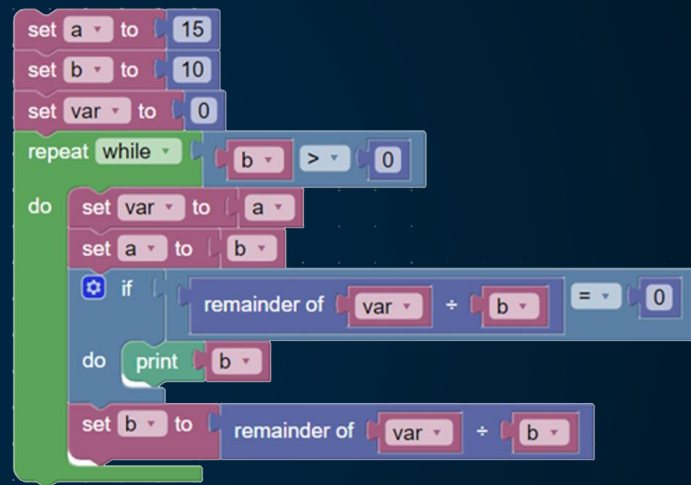
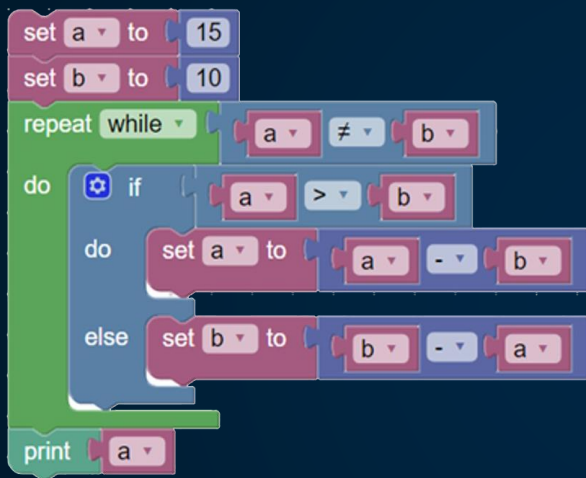




گاهی اوقات اضافه کردن برخی شرطها به الگوریتم باعث بهینه‌تر شدن آن می‌شود. اما گاهی اوقات لازم است که خود الگوریتم تغییر پیدا کند تا پاسخ مسئله در زمان کمتر و یا با استفاده‌ی کمتری از حافظه پیدا شود.

چه الگوریتم دیگری برای ب.م.م. پیشنهاد می‌دهید؟

در تصویر زیر الگوریتم دیگری به نام **Euclidean Algorithm** (الگوریتم اقلیدس) در دو شیوه برای پیدا کردن ب.م.م. نوشته شده است.



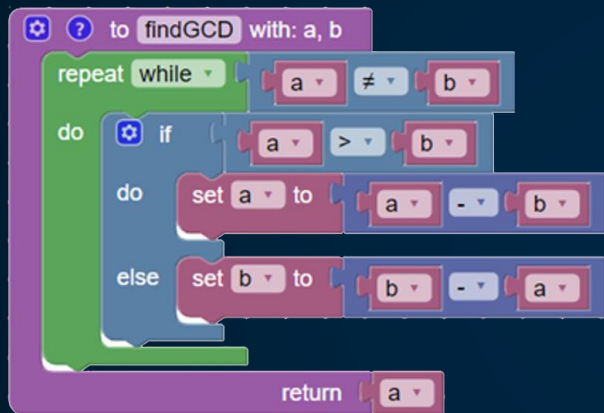
به نظر شما تفاوت این ۲ شبه‌کد چیست؟



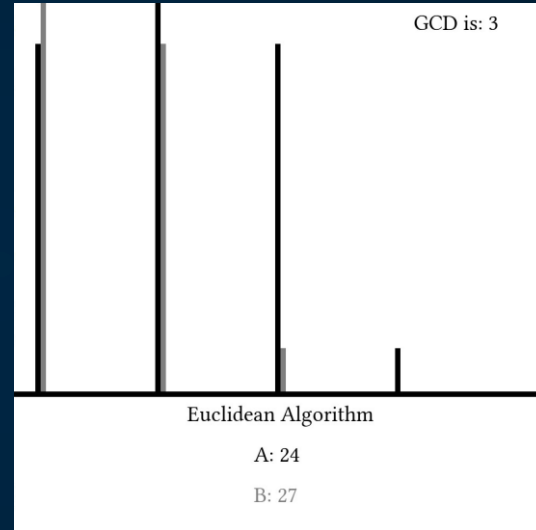
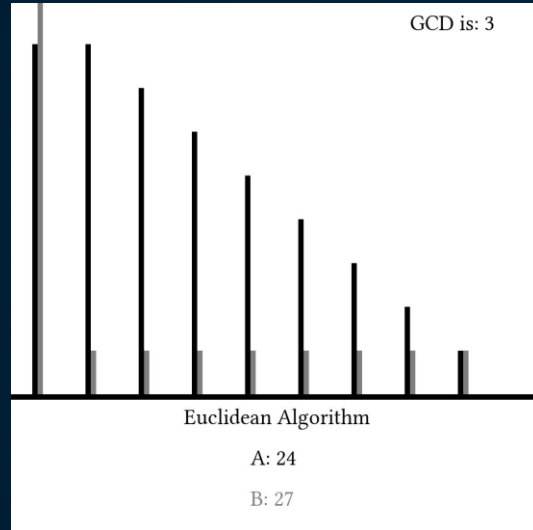
آنها را با هم مقایسه کنید و مزایا و معایب هر کدام را بررسی کنید. به نظر شما هر کدام در چه شرایطی عملکرد بهتری دارند؟

در صورتی که الگوریتم‌های بالا را به صورت تابع‌های جداگانه بنویسیم، بهتر می‌توانیم عملکرد آن دو را مقایسه کنیم.

در شکل یکی از الگوریتم‌ها به صورت تابع نوشته شده است. سعی کنید با هم‌گروهی خود الگوریتم دیگر را به صورت تابع بنویسید.



مراحل اجرای دو الگوریتم را برای دو عدد ۲۴ و ۲۷ در نمودارهای زیر بررسی کنید. 



```
to findGCD with: a, b
  repeat while a ≠ b
    do if a > b
      do set a to a - b
    else
      set b to b - a
  return a
```

```
set a to 15
set b to 10
print findGCD with:
  a
  b
```



به نظر شما آیا این مسئله راه حل دیگری دارد؟ 

اگر دوست داشتید برای درک بهتر این الگوریتمها و تفاوت میان آنها می توانید از لینک های زیر کمک بگیرید.



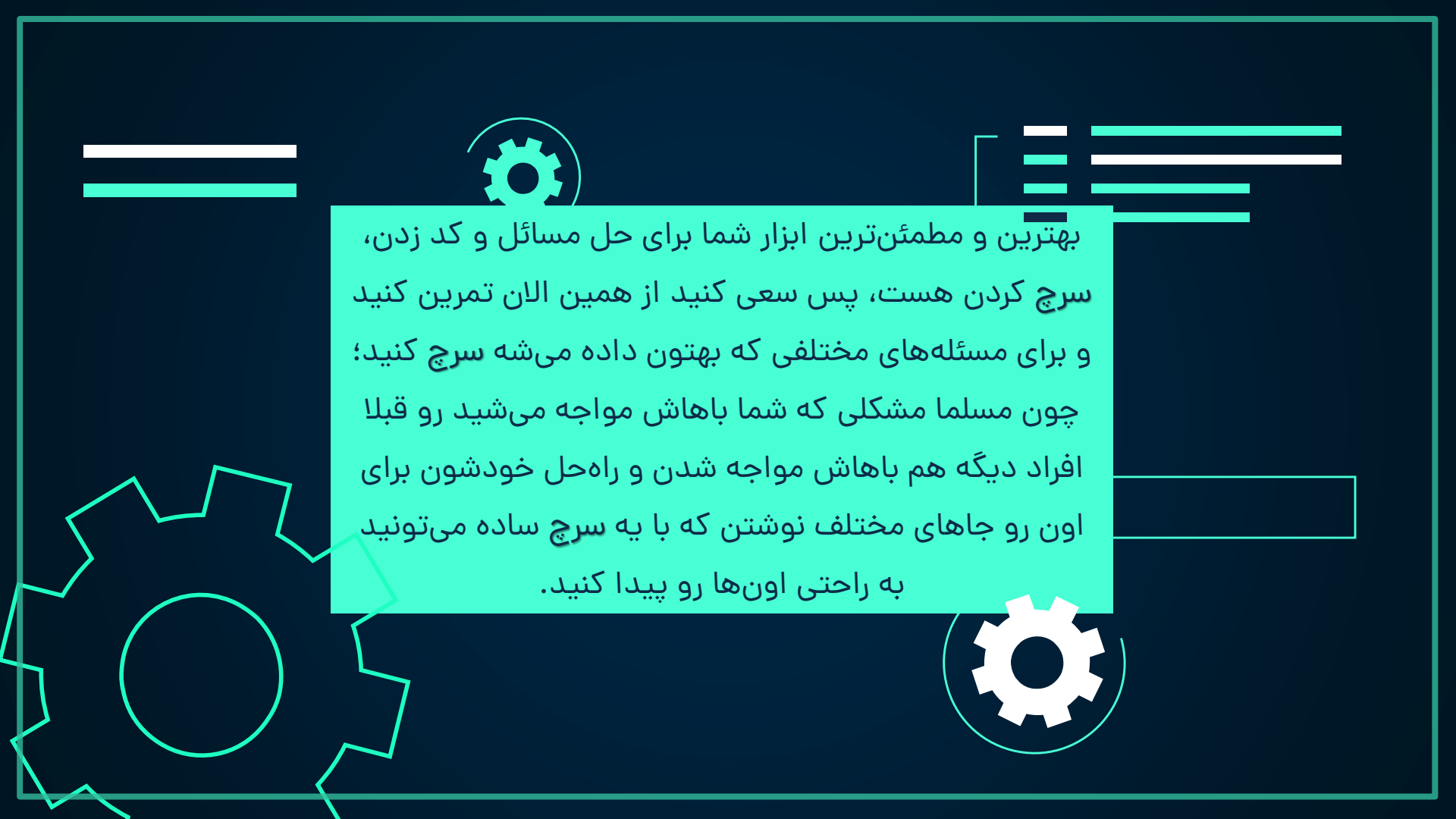
[The Euclidean Algorithm \(Khan Academy\)](#)



[The Euclidean Algorithm \(algorithm-archive\)](#)




[The Euclidean Algorithm \(youtube\)](#)





بهترین و مطمئن‌ترین ابزار شما برای حل مسائل و کد زدن،  
سرچ کردن هست، پس سعی کنید از همین الان تمرین کنید  
و برای مسئله‌های مختلفی که بهتون داده می‌شه سرچ کنید؛  
چون مسلماً مشکلی که شما باهاش مواجه می‌شید رو قبلاً  
افراد دیگه هم باهاش مواجه شدن و راه‌حل خودشون برای  
اون رو جاهای مختلف نوشتن که با یه سرچ ساده می‌تونید  
به راحتی اون‌ها رو پیدا کنید.





## سوال دوم: ساعت

حلقه‌ها در برنامه‌نویسی به چه معنا هستند و به چه علت استفاده می‌شوند؟  
چه مفاهیم و اتفاقاتی در دنیای اطراف همواره در حال تکرار شدن هستند؟ 

ساعت را می‌توان به عنوان یکی از وسایلی دانست که دنیای آن به حلقه یا loop محدود شده است. به نظر شما آیا یک حلقه برای در دست گرفتن زمان کافی است؟ 


اگر یک حلقه داخل حلقه‌ای دیگر استفاده شود به آن **حلقه‌ی تو در تو** یا **nested loop** گفته می‌شود. در این نوع حلقه‌ها به ازای اجرای هر بار حلقه بیرونی، حلقه داخلی به‌طور کامل انجام می‌شود. 


ساعت نمونه‌ی خوبی است که مانند حلقه‌های تو در تو عمل می‌کند. به این صورت که برای یک حرکت عقربه ساعت‌شمار به عدد بعدی لازم است تا عقربه‌ی دقیقه‌شمار یک دور کامل بچرخد. 

حال از شما می‌خواهیم که در گروه خود با توجه به توضیحات بالا شبه‌کدی برای شبیه‌سازی یک ساعت دیجیتال بنویسید که ساعت، دقیقه و ثانیه را برای یک شبانه روز کامل چاپ کند. 

آیا می‌توانید تعداد دفعات اجرای درونی‌ترین حلقه را محاسبه کنید؟ 

# سوال سوم: مبنا

به نظر شما چه اتفاقی می افتاد اگر ما به جای ۱۰ رقم برای نشان دادن اعداد، ۷ رقم داشتیم؟ آیا در سیستم جدید برخی اعداد وجود نخواهند داشت؟ 

برای جواب به این سوال، به طور کلی می توان گفت که **اعداد** تغییری نخواهند کرد و تنها شیوهی نمایش آنها توسط **ارقام**، متفاوت خواهد شد. به عنوان مثال از این به بعد به جای عدد هشت باید ۱۱ می نوشتیم. (چرا؟) 

به این کار در دنیای ریاضی **تغییر مبنا** گفته می شود. 




تغییر مبنا در علم ریاضی و مخصوصا کامپیوتر - از آنجایی که دنیای کامپیوتر، دنیای صفر و یک‌هاست و می‌توان گفت همه چیز در این دنیا در مبنای ۲ قرار دارد - بسیار کاربردی و مهم است. اما شاید برای ما که همیشه با دنیای دهمی سر و کار داشته‌ایم، این تغییر مبنا کمی وقت‌گیر باشد.




برای حل این مشکل، می‌توانیم برای تغییر مبنا یک شبه‌کد آماده کنیم تا راحت‌تر بتوانیم تشخیص دهیم که اعداد در یک سیستم جدید، از چه ارقامی تشکیل شده‌اند. در این شبه‌کد قرار است دو عدد طبیعی، ورودی‌های ما باشند. اگر نام آن‌ها را  $a$  و  $b$  بگذاریم، می‌خواهیم خروجی شبه‌کد ما،  $(a)_b$  یعنی  $a$  در مبنای  $b$  باشد تا دیگر برای تبدیل اعداد در مبناهای مختلف با مشکلی مواجه نشویم.

# اما سوال آخر: عددبازی

قبل از این که بریم سراغ سوال آخر، لازمه که خبری رو اعلام کنیم... 

دو نفر از برترین برنامه‌نویس‌های دنیا (کدخدا و Botfather) قصد دارن به دنیا ثابت کنن که می‌شه هر چیزی رو در دنیا به کد تبدیل کرد. اون‌ها برای رسیدن به این هدف، لازم دارن که یک تیم قوی از برنامه‌نویس‌ها و مهندسين کامپیوتر رو جمع آوری کنن. برای همین بخشی از دستورکار کارگاه‌ها رو در اختیار گرفتن تا بتونن افراد بیش‌تری رو به گروه خودشون جذب کنن و توی این مسیر تحولی در روند آموزش هم ایجاد کنن. 

به دلیل اهمیت بالای شبه‌کد و توانایی درک الگوریتم، اون‌ها این جلسه از کارگاه‌های مبانی رو به عنوان شروع کار خودشون انتخاب کردن. ادامه‌ی ماجرا رو از زبان خودشون می‌شنویم تا ببینیم برای این جلسه چه تمرینی رو برای شما آماده کردن. 



سلام به همه... من Botfather هستم. خیلی خوشحالم که همراه کدخدا می‌تونیم این ترم در کنار شما باشیم.



من هم به همگی سلام عرض می‌کنم. خیلی وقتتون رو نمی‌گیرم. بریم سراغ دستورکار...



برای شروع کار امروزمون، اول از همه اجازه بدید شما رو با دوست قدیمی و عزیزم، Numfather، آشنا کنم. Numfather پدر بازی عددبازی است و سالیان ساله که داره این بازی رو بین طرفدارانش برگزار می‌کنه.



این بازی از اعداد ۱ تا ۱۰ و دو دستور کلی تشکیل شده که بازیکن می‌تونه در هر بار نوبتش هر کدوم از اون‌ها رو انتخاب کنه. کار Numfather پخش کردن نامحدود اعداد به صورت تصادفی و انجام دستورات بازیکن در طول بازیه.



بازی این طوری شروع می‌شه که Numfather دو عدد تصادفی به بازیکن می‌ده، دو عدد تصادفی هم برای خودش برمی‌داره و یکی از اعدادش رو باز هم به صورت تصادفی اعلام می‌کنه. از این به بعد بازیکن باید با دستوراتش بازی رو پیش بیره تا به برد نزدیک بشه. حالا این دستورات چی هستن و شرط پیروزی چیه؟



برنده بازی با مقایسه مجموع اعداد Numfather و بازیکن مشخص می‌شه. شرایط برد و باخت به این صورته:

۱. برنده فردیه که مجموع اعدادش ۲۱ بشه.
۲. اگر این جمع برای فردی از عدد ۲۱ گذشته باشه بازنده محسوب می‌شه.
۳. اگر هیچ کدوم از این شرایط اتفاق نیفته، باز هم مجموع عددها تعیین‌کننده است طوری که اگه حاصل آن برای هر دو بازیکن برابر باشه بازی مساویه و در غیر این صورت فردی که به ۲۱ نزدیک‌تره برنده‌ی بازی می‌شه.



و اما دستورات بازیکن:

**AddAdad** به این معناست که بازیکن می‌خواهد یک عدد جدید به مجموع اعدادش اضافه کند و Numfather یک عدد تصادفی بهش تحویل می‌دهد.

**AdadBas** دستوره که بازیکن در انتهای بازی اعلام می‌کند. در واقع زمانی که بازیکن با مجموع اعدادش فکر می‌کند شانس برنده شدن رو داره و عدد جدیدی نمی‌خواهد، این دستور رو می‌دهد و پس از اون طبق قوانین بازی به اعداد Numfather تا سقف مجموع ۱۷، عدد تصادفی اضافه می‌شه تا مقایسه‌ی نهایی انجام و برنده مشخص شه.



داستان این جلسه ما از ایمیلی که به تازگی از Numfather به دست ما رسیده شروع می‌شه. متن ایمیل رو بخونید تا متوجه بشید داستان از چه قراره.



به نام خالق اعداد

کدخدا و botfather عزیز

در ابتدا شروع فعالیت آموزشی جدیدتان را تبریک می‌گویم و برای‌تان آرزوی موفقیت دارم. پس از آن درخواستم را مطرح می‌کنم و نیازمند یاری سبزتان هستم. چه کسی از شما بهتر می‌داند که این روزها تمام دنیا عدد شده است و خارج از دنیای صفر و یک شما، اعداد بسیار زیادی برای رسیدگی وجود دارند که کار مرا به شدت دشوار کرده‌اند. از طرفی این دوست قدیمی‌تان در حال رفتن رو به کهنسالی است و چه کار بهتر از آن که مسئولیت‌هایش را به کامپیوترهای دقیق شما بسپارد؟ از آن‌جا که به تازگی در برگزاری عددبازی به مشکل خورده‌ام از شما تقاضا دارم برنامه‌ای به این منظور طراحی کنید تا طرفداران این بازی همچنان بتوانند از آن لذت ببرند. در ادامه، شبه‌کدی که سعی کرده‌ام با توجه به وظایفم در بازی بنویسم را ارسال می‌کنم و وقت آن است که کار را به کاردان سپرده و از شما برای اتمام آن کمک بگیرم. از لطف بی‌دریغتان سپاسگزارم.

دوستدارتان Numfather

```
set father_hand to RANDOM(1,10)
```

```
set your_hand to RANDOM(1, 10) + RANDOM(1, 10)
```

```
set adadBas to zero
```

```
while father_hand < 21 and your_hand < 21 and adadBas is zero:
```

...

```
endwhile
```

```
print father_hand
```

```
print your_hand
```

```
if father_hand = your_hand:
```

```
    print "It's a tie game!"
```

ادامه‌ی  
شبه‌کد

```
else if father_hand = 21 or your_hand > 21:
```

```
    print "Father wins!"
```

```
else if your_hand = 21 or father_hand > 21:
```

```
    print "You win!"
```

```
else if father_hand > your_hand:
```

```
    print "father wins!"
```

```
else:
```

```
    print "You win!"
```



همون‌طور که متوجه شدید قراره شبه‌کد Numfather رو کامل کنیم. لازمه یک سری توضیحات رو راجع به این شبه‌کد بدم تا قبل از شروع کار براتون ابهامی وجود نداشته باشه. اولین نکته اینه که متغیر adadBas برای مشخص کردن انتخاب یا عدم انتخاب این دستور توسط بازیکنه و هنگام اجرای این دستور باید مقدار این متغیر تغییر کنه. به عنوان دومین نکته فراموش نکنید که Numfather یک عدد مخفی دیگه هم داره که باید در پایان بازی به اعدادش اضافه بشه.



من هم از شما می‌خوام تا به این مساله فکر کنید که اگر قرار باشه بازی رو به نحوی پیاده سازی کنیم تا پس از اتمام هر دور و مشخص شدن برنده، بازیکن بتونه دور جدیدی رو شروع کنه، به اعمال چه تغییراتی در شبه‌کد نیاز داریم؟

امیدوارم به خوبی از پس این چالش بربیاید، در آینده‌ی نزدیک هم بتونید شبه‌کد خودتون رو به کد تبدیل کنید و از این بازی لذت ببرید. موفق باشید!



من هم براتون آرزوی موفقیت می‌کنم. تا دستورکار بعدی خدا نگهدار ☺

;