به نام خدا

**برای هر پروژه سه گروه می‌توانند ثبت نام کنند. ممکن است چند پروژه دیگر اضافه شوند.**

**پروژه‌ی اول:** فرض کنید یک بزرگراه داریم که توسط یک معادله‌ی خطی مشخص شده است. همچنین، تعدادی انبار در کنار بزرگراه وجود دارند که اجناس موجود در آنها را باید در نقطه‌ای کنار بزرگراه جمع کنیم تا از طریق بزرگراه به شهرهای دیگر منتقل شوند. هدف مساله پیدا کردن یک نقطه روی بزرگراه است که مجموع فاصله‌ی آن از تمام انبارها حداقل باشد. اولین ورودی وارد شده توسط کاربر تعداد انبارها است. سپس، سه عدد اعشاری (A، B و C) توسط کاربر داده می‌شود که مختصات بزرگراه را به صورت Ax+By+C=0 مشخص می‌کند. سپس مختصات دکارتی انبارها وارد می گردد. اگر A، B و C طوری داده شود که معادله یک خط را مشخص نکنند (مثلاً A و B برابر صفر باشند)، باید یک پیغام خطای مناسب تولید شود. خروجی برنامه مجموع فاصله‌ی انبارها از نقطه‌ی بهینه است. به ورودی زیر توجه کنید:

1

5

3 -5 -7

1 3

-2 4

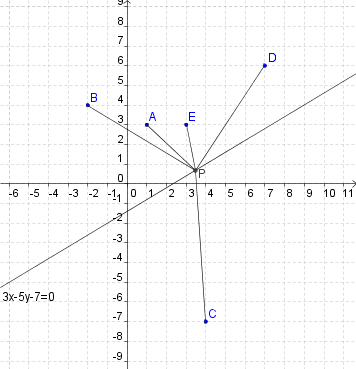
4 -7

7 6

3 3

خروجی:

26.232349



**گروه اول:** نعمت قمری-محسن صائب گروه دوم: رضاپسند

**پروژه‌ی دوم:** فرض کنید که n نقطه در مختصات دکارتی داریم. مختصات نقاط از اعداد طبیعی تشکیل شده است. با این نقاط می‌توان چهارضلعی‌های مختلفی درست کرد. می‌گوییم که چهارضلعی ABCD خوش تعریف است اگر:

* Ax > 0 و Ay > 0;
* Bx > 0 و By < 0;
* Cx < 0 و Cy < 0;
* Dx < 0 و Dy > 0;
* مساحت ABCD یک عدد طبیعی باشد.

وظیفه‌ی شما این است که تعداد چهارضلعیهای خوش تعریف را پیدا کنید. در ورودی، ابتدا تعداد نقاط داده می‌شود. سپس، مختصات نقاط وارد می‌شود. نقاط وارد شده نباید مثل هم باشند. به عنوان مثال برای ورودی زیر خروجی برابر ۲ است:

6

1 1

2 2

-1 -1

-2 2

2 -1

-3 -4

**پروژه‌ی سوم:** در این پروژه، تعداد خطوط کد در یک فایل جاوا را می‌شماریم. شما در هنگام شمارش باید از خطوط خالی و توضیحات صرفنظر کنید. دقت کنید که هر خط ممکن است شامل چندین توضیح به صورت /\* … \*/ باشد. توضیحات /\* … \*/  نمی‌توانند تودرتو باشند. و ممکن است یک رشته شامل توضیح نیز باشد. به مثالهای زیر توجه کنید:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | - // This file contains 3 lines of code  1 public interface Dave {  - /\*\*  - \* count the number of lines in a file  - \*/  2 int countLines(File inFile); // not the real signature!  3 } |

و ...:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | - /\*\*\*\*\*  - \* This is a test program with 5 lines of code  - \* \/\* no nesting allowed!  - //\*\*\*\*\*//\*\*\*/// Slightly pathological comment ending...  -  1 public class Hello {  2 public static final void main(String [] args) { // gotta love Java  - // Say hello  3 System./\*wait\*/out./\*for\*/println/\*it\*/("Hello/\*");  4 }  -  5 } |

**گروه اول:** موسی نژاد – زینی وند

**پروژه‌ی چهارم:** فایل [football.dat](http://codekata.com/data/04/football.dat) شامل نتایج لیگ برتر انگلیس در سال ۲۰۰۱ است. ستونهای F و A به ترتیب تعداد گلهای زده و خورده هر تیم را نشان می‌دهد. مثلاً، آرسنال ۷۹ گل زده و ۳۹ گل خورده است. برنامه‌ای بنویسید که تیمها را به ترتیبهای تفاضل گل، گل زده، گل خورده، تعداد برد، تعداد باخت مرتب کند.

**گروه اول**: وطن دوست، ارکوازی **گروه دوم**: تیباش **گروه سوم**: فرزانه عبداللهی

**پروژه‌ی پنجم:** در این پروژه فیلتر بلوم را پیاده می‌کنید. فیلتر بلوم به درد پایگاه داده‌های خیلی بزرگ می‌خورد و مشخص می‌کند که آیا داده‌ی مورد نظر در پایگاه داده موجود است یا نه. اما مشکلی که دارد این است که ممکن است فیلتر اعلام کند یک داده در پایگاه داده موجود است در حالی که این گونه نیست. البته بر عکس آن امکان پذیر نیست. یعنی، اگر فیلتر بگوید که داده در پایگاه داده موجود نیست قطعاً درست خواهد بود. نکته‌ی بعدی این که می توانید دقت فیلتر را کنترل کنید.

فیلتر بلوم دارای یک آرایه از بیتها است که در ابتدا همه‌ی آنها برابر صفر هستند. هر بار که یک داده را وارد پایگاه داده کنیم، داده را به یک تابع درهم‌سازی Hash می‌دهیم تا یک عدد تولید شود. حال، تمام بیتهای متناظر با عدد در آرایه را برابر “1” تنظیم می‌کنیم. اگر بخواهیم وجود یک داده در پایگاه داده را چک کنیم بدین صورت عمل می کنیم که Hash داده را محاسبه می‌کنیم. اگر بیتهای متناظر در آرایه تنظیم شده بودند به احتمال خیلی زیاد داده در پایگاه داده موجود است. اگر هنگام ورود داده به پایگاه داده، Hash آن طوری باشد که تمام مکانهای متناظر قبلاً تنظیم شده باشند می گوییم که برخورد رخ داده است. مشکل عدم دقت فیلتر بلوم همین امکان وقوع برخورد است. کلمات وارد شده در پایگاه داده از نوع رشته هستند. لطفاً برای مشخص کردن تابع Hash به اینجانب مراجعه کنید.

**پروژه‌ی ششم:** عدد ۴۷ را در نظر بگیرید. اگر ۴۷ را با معکوس خودش جمع کنیم، یعنی ۴۷+۷۴=۱۲۱ یک عدد دوجهته به دست می‌آید. البته بعضی اعداد نمی‌توانند خیلی سریع تبدیل به اعداد دو جهته شوند. به عنوان مثال:

349 + 943 = 1292,  
1292 + 2921 = 4213  
4213 + 3124 = 7337

یعنی عدد ۳۴۹ بعد از سه تکرار تبدیل به یک عدد دوجهته می گردد. البته، تصور بر این است که بعضی اعداد مثل ۱۹۶ هرگز نمی‌توانند تبدیل به یک عدد دوجهته شوند. البته این موضوع ثابت نشده است. به اعدادی که هیچگاه طی فرآیند معکوس کردن و جمع زدن تبدیل به یک عدد دوجهته نمی شوند، عدد لیچرل گفته می‌شود. هم به خاطر طبیعت نظری این اعداد و هم به خاطر این که مساله‌ی زیر قابل حل باشد، ما فرض می کنیم همه‌ی اعداد لیچرل هستند مگر این که خلاف آن ثابت شود. به علاوه، برای تمام اعداد کوچکتر از ۱۰۰۰۰، یا عدد مورد نظر بعد از ۵۰ تکرار تبدیل به یک عدد دوجهته می‌گردد، یا این که هیچکس با قدرت کامپیوترهای امروزی نوانسته است ثابت کند که می توان این عدد را به یک عدد دوجهته تبدیل کرد. در واقع، ۱۰۶۷۷ اولین عددی است که ثابت شده است می‌توان آنرا با بیش از ۵۰ تکرار (دقیقا ۵۳ تکرار) به یک عدد دوجهته 4668731596684224866951378664 تبدیل کرد. نکته ی جالب دیگر این است که بعضی اعداد دوجهته (مثل ۴۹۹۴) خودشان لیچرل هستند.

سوال این است: تعداد اعداد لیچرل کمتر از ۱۰۰۰۰ چند تا است؟

**گروه اول:** یاسمن دستان**- گروه دوم:** رضا بهره- محمد امین چنانی **گروه سوم**: غلامحسینی-میلاد منصوری

**پروژه‌ی هفتم:** می‌خواهیم وابستگی بین کلاسهای مختلف یک پروژه را حساب کنیم. فرض کنید که A به کلاس B وابسته باشد. اگر B به کلاسهای X و Y وابسته باشد، A نیز به Y و X وابسته است. یعنی خاصیت تعدی برقرار است. به عنوان مثال، ورودی زیر را در نظر بگیرید:

A B C

B C E

C G

D A F

E F

F H

برنامه باید خروجی زیر را تولید کند:

A B C E F G H

B C E F G H

C G

D A B C E F G H

E F H

F H

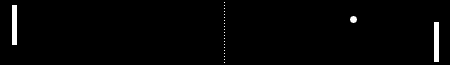
**گروه اول:** محمد جواد پروانه **گروه دوم:**

**پروژه هشتم:** اگر از عدد ۱ شروع کنیم و در جهت خلاف عقربه‌های ساعت بچرخیم و اعداد را بنویسیم، مربع زیر تولید می‌گردد:

**37** 36 35 34 33 32 **31**  
38 **17** 16 15 14 **13** 30  
39 18  **5**  4  **3** 12 29  
40 19  6  1  2 11 28  
41 20  **7**  8  9 10 27  
42 21 22 23 24 25 26  
**43** 44 45 46 47 48 49

طول این مربع برابر ۷ است. همانطور که می‌بینید اعداد اول روی قطر قرار دارند. از ۱۳ عددی که روی دو قطر وجود دارند، ۸ عدد یعنی ۶۲٪ اول هستند. اگر چرخش را ادامه دهیم، طول مربعی که میزان اعداد اول روی دو قطر آن کمتر از ۱۰٪ است چند است؟

**پروژه نهم:** در این پروژه یک بازی پینگ پونگ ساده مثل شکل زیر را طراحی کنید:



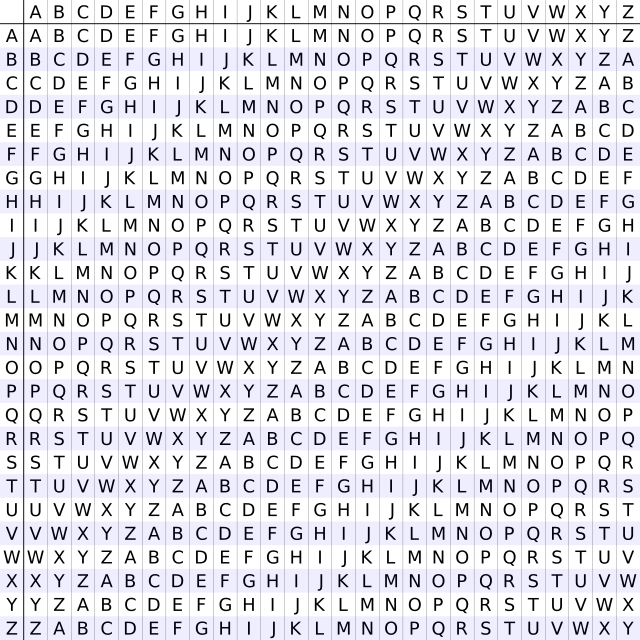
هر کدام از بازیگرها از دو کلید برای حرکت به بالا و پایین استفاده می کنند. به کاربر اجازه دهید که اندازه‌ی راکت و طول و عرض زمین را خودش تنظیم کند.

**گروه اول**: طیبه ی شریعتی حورا هیوری **گروه دوم**: زهره احمدی **گروه سوم**: نازنین ایازی- الهام رمضانی

**پروژه دهم:** در این پروژه روش رمزنگاری ویژنر را پیاده می‌کنید. فرض کنید می‌خواهیم پیغام ATTACKATDAWN را رمزگذاری کنیم. ابتدا باید یک کلید انتخاب کنیم. فرض کنید کلید انتخاب شده برابر LEMON باشد. از آنجا که طول پیغام و طول کلید متفاوت است، کلید را با استفاده از تکرار گسترش می دهیم تا با پیغام هم اندازه شود. پس، کلید برابر خواهد بود با: LEMONLEMONLE. برای رمزگذاری، از یک جدول رمز استفاده می‌شود که یک نمونه از آن در زیر داده شده است.

در این جدول، الفبای انگلیسی به ۲۶ حالت مختلف مرتب شده اند. ما یک حرف از پیغام و یک حرف از کلید را انتخاب می کنیم و با استفاده از جدول آنرا رمز می‌کنیم. مثلاً، حرف اول پیغام برابر A و حرف اول رمز برابر L است. در جدول، اگر ستون A و ردیف L (یا ردیف A و ستون L ) را جستجو کنیم به حرف L می‌رسیم. برای حروف دوم پیغام و کلید یعنی T و E، رمز X به دست می آید و .... اگر همینطور ادامه دهیم پیغام رمز شده با LXFOPVEFRNHR برابر خواهد بود.

پیغام اصلی در یک فایل قرار دارد و کلید توسط کاربر وارد می‌گردد. خروجی پروژه یک فایل رمز شده است. همچنین، شما باید بتوانید یک فایل رمز شده را به حالت اولیه برگردانید.

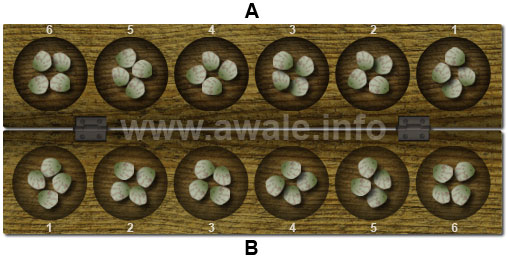


**گروه اول:** قائدی **گروه دوم**: مجید و مجتبی فاضلی **گروه سوم**: جانیان

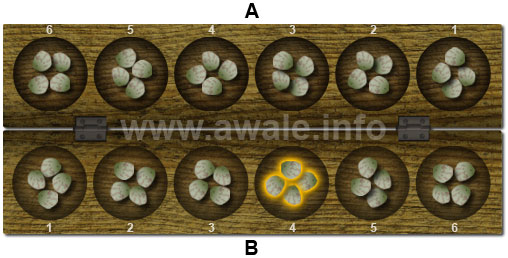
**پروژه یازدهم:** در این پروژه بازی بیست سوالی در مورد حیوانات اجرا می گردد. شما لیست حیوانات را به کاربر نشان می‌دهید و از کاربر می‌خواهید که یکی از حیوانات را در نظر بگیرد. سپس، از کاربر تعدادی سوال می‌پرسید که جواب آنها بله یا خیر است. با توجه به جوابهای کاربر، اسم حیوان را حدس می‌زنید. حداقل تعداد حیوانات ۱۵ است و شما حداقل باید ۳۰۰ سوال مختلف داشته باشید. برای پیاده سازی صحیح این پروژه به مفهوم درخت باینری نیاز خواهید داشت.

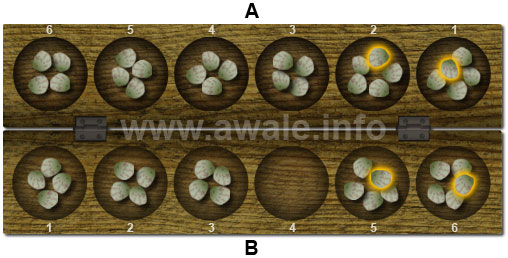
**گروه اول**: سمیر منفرد

**پروژه دوازدهم:** بازی آواری یکی از بازیهای بومیان آفریقا است که از طریق سنگ و سطل انجام می‌شود. این **بازی شامل ۱۲** سطل و ۴۸ سنگ است. سطلهای ۱ تا ۶ مال بازیکن A و سطلهای 7 تا ۱۲ مال بازیکن B است. هدف این بازی این است که سنگهای بیشتری جمع‌آوری شود. اگر بازیکنی بیشتر از ۲۴ سنگ جمع کند می‌تواند پیشنهاد خاتمه ی بازی را بدهد. یا می توان بازی را تا وقتی که نتوان سنگ دیگری را خارج کرد ادامه داد. در ابتدا هر کدام از سطلها دارای ۴ سنگ است.



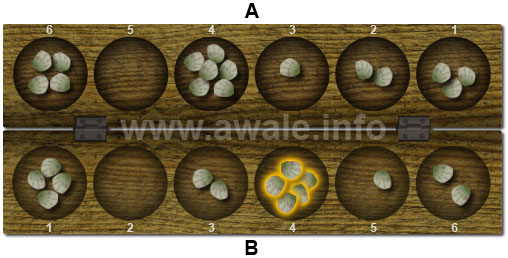
**قانون اول:** هر بار که نوبت به یک بازیگر رسید، آن بازیگر یکی از خانه های خود را انتخاب می کند و سنگهای آنرا بر می‌دارد و آنها در جهت خلاف حرکت عقربه های ساعت در سطلها بخش می کند. به هر سطل فقط یک سنگ اضافه می‌شود و در صورتی که سطلهای خودمان تمام شود از سطلهای حریف استفاده می کنیم. به شکل زیر دقت کنید:

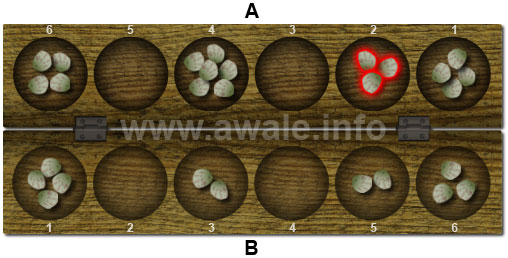
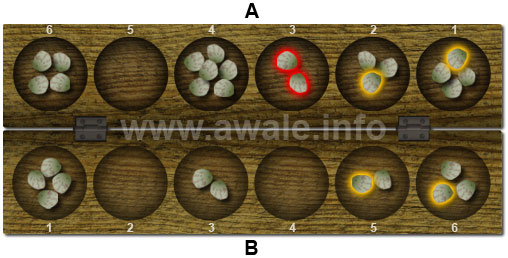




در اینجا، بازیگر B سطل ۴ را انتخاب و سنگهای آنرا پخش کرده است.

**قانون دوم**: پس از این که بازیگر سنگهای خود را پخش نمود، اگر آخرین خانه ای که به آن سنگ اضافه کرده است متعلق به حریف بود و تعداد سنگهای ۲ یا ۳ بود، آن سنگها را برای خود بر می‌دارد. سپس، به خانه ی قبلی نگاه می‌کند. اگر تعداد سنگهای این سطل هم ۲ یا سه بود آنها را بر می‌دارد. اینکار را آنقدر تکرار می کنیم که به خانه ای برسیم که تعداد سنگهای ۲ یا ۳ نباشد. به شکل زیر دقت کنید:





**قانون سوم**: اگر سطلی که سنگهای آنرا برداشته‌اید دوازده سنگ یا بیشتر داشته باشد، در اینصورت می‌توان یک دور در تمام خانه ها سنگ قرار داد. اما حق ندارید سنگ دوازدهم را در سطل خالی شده بگذارید و از آن سطل صرفنظر کنید.

**قانون چهارم**: حق ندارید در یک حرکت تمام سطلهای حریف را خالی کنید. انجام اینکار به منزله ی قبول شکست از طرف شما است.

**قانون پنجم**: اگر ادامه ی بازی عاقلانه نباشد یا امکان برداشتن سنگ دیگری نباشد، دو طرف می‌توانند بازی را خاتمه دهند و کسی که سنگ بیشتری جمع کرده است برنده است.

**گروه اول:** بهنام مرادی – حمزه معاضدی **گروه دوم**: رشید نجفی

**پروژه سیزدهم:** در این بازی کامپیوتر یک عدد n رقمی (2<n<7) را در نظر می‌گیرد که باید کاربر آنرا حدس بزند. تعداد ارقام عدد از قبل توسط کاربر تعیین می‌گردد. همچنین، عدد دارای رقم تکراری نیست. کاربر یک حدس را وارد می‌کند. اگر هیچکدام از ارقام حدس کاربر درست نباشد، یک چراغ قرمز روشن می‌شود. اگر کاربر یک رقم را درست حدس بزند ولی جای آن غلط باشد، یک چراغ زرد دریافت می‌کند. اگر کاربر رقم و جای آن را درست حدس زده باشد، یک چراغ سبز دریافت می‌نماید. به عنوان مثال، اگر عدد کامپیوتر ۱۲۳۴ و حدس کاربر ۲۵۱۴ باشد، کاربر یک چراغ سبز و دو چراغ زرد دریافت می‌کند. بازی آنقدر ادامه پیدا می‌کند تا کاربر بتواند عدد را حدس بزند.

**گروه اول**: میلاد ایازی - شیرویی **گروه دوم**: شعبانی – ولیان **گروه سوم**: نجف زاده- اکبری پور