**به نام خدا**

**تکلیف تحویلی سری 1 درس برنامه‌سازی پیشرفته آخرین مهلت تحویل: 15/1/1404**

**هدف:** این سری تکلیف تحویلی مشتمل بر مسائلی ساده جهت حل و پیاده‌سازی در زبان برنامه نویسی Java است. انجام این تکلیف به شما صرفا در مدیریت تحلیل، دستورات و ساختارهای کنترلی اصلی زبان برنامه نویسی Java ، فارغ از مفاهیم مرتبط با شئ‌گرایی کمک کرده و به مدرس امکان ارزیابی مهارت‌های دانشجویان در برنامه‌نویسی در محدوده‌ی این مفاهیم را خواهد داد.

توضیحات لازم:

1. نوشتن توضیحات[[1]](#footnote-1) حداقلی درکدها، علی‌الخصوص در مواردی که برای فهم عملکرد کد نیاز به توضیح هست ضروری می‌باشد.
2. این تکلیف باید حداکثر تا آخرین مهلت و ساعت تعیین شده **فقط** از طریق سامانه‌ی VU ارسال شود.
3. فایلی که جهت تحویل ارسال می شود باید به صورت یک فایل فشرده‌ی .zip باشد، به صورتی که نام فایل فشرده‌ متشکل از نام و نام خانوادگی دانشجو با حروف انگلیسی است. مثلا فایل فشرده‌ی دانشجوی فرضی با نام علی صادقی نامش AliSadeghi.zip می‌باشد. درون این فایل فشرده بایستی تعدادی فایل با پسوند .java باشند که هر یک شامل شامل نام و نام خانوادگی دانشجو با حروف انگلیسی، خط تیره به همراه شماره‌ی سوال می‌باشند. مثلا نام فایل کدی دانشجوی فرضی مذکور بایستی AliSadeghi-Q1.java باشد.
4. در صورت مشاهده‌ یا احراز احتمالی هرگونه تقلب ولو در یک سوال، از قبیل کپی‌برداری یک کد از اینترنت، کتب، از سایر دانشجویان، استفاده از ابزارهای هوش مصنوعی و یا نوشتن کد توسط شخصی به جز فرد، کل تکلیف تحویلی مربوطه تصحیح نخواهد شد.

**...................................................................................................................................................................................**

1. (*چاپ الگو*) با استفاده از forهای تودرتو برنامه‌ای بنویسید که ساعت شنی زیر را تولید کند:

|""""""""""|

\::::::::/

\::::::/

\::::/

\::/

||

/::\

/::::\

/::::::\

/::::::::\

|""""""""""|

1. (*دمای خنکی باد*) بیرون چقدر سرد است؟ دما به تنهایی برای ارائه‌ی پاسخ کافی نیست. عوامل دیگری از جمله سرعت باد، رطوبت نسبی، و نور خورشید نقش مهمی در تعیین سرمای بیرون دارند. در سال 2001، سرویس ملی هواشناسی (NWS) برای اندازه‌گیری سرما با استفاده از دما و سرعت باد، دمای جدیدی موسوم به خنکی باد را پیاده‌سازی کرد که فرمولش بدین صورت است:

که در آن *ta* دمای بیرونی بر حسب درجه فارنهایت، *v* سرعت بر حسب مایل در ساعت و *twc* دمای خنکی باد است. فرمول بالا نمی‌تواند برای سرعت باد کمتر از 2 مایل در ساعت یا دمای زیر 58- فارنهایت یا بالاتر از 41 درجه فارنهایت استفاده شود.

برنامه‌ای بنویسید که از کاربر بخواهد دمایی بین 58- درجه فارنهایت و 41 درجه فارنهایت و سرعت بادی بزرگتر یا مساوی 2 را وارد کرده و سپس دمای خنکی باد را نشان دهد.

1. (*روز هفته*) تناسب Zeller الگوریتمی است که توسط Christian Zeller برای محاسبه‌ی روز هفته توسعه یافته است. فرمول از این قرار است:

که در آن

* h روز هفته است (0: شنبه، 1: یکشنبه، 2: دوشنبه، 3: سه شنبه، 4:چهارشنبه، 5: پنجشنبه، و 6: جمعه).
* q روز ماه است.
* m ماه است (3: مارس، 4: آوریل، ...، 12: دسامبر). ژانویه و فوریه ماه‌های 13 و 14 از سال قبل محسوب می‌شوند.
* j برابر است.
* k سال قرن است (یعنی، *year* % 100).

توجه داشته باشید که تمام تقسیمات در این تمرین یک تقسیم عدد صحیح انجام می‌دهند. برنامه‌ای بنویسید که از کاربر می‌خواهد سال، ماه و روز ماه را به میلادی وارد کرده و نام روز هفته را نمایش دهد.

(*راهنمایی*: ژانویه و فوریه به عنوان 13 و 14 در فرمول حساب می‌شوند، بنابراین شما باید برای ماه ورودی 1 کاربر را به 13 و 2 را به 14 تبدیل کنید و سال را به سال قبل تغییر دهید. به عنوان مثال، اگر کاربر 1 را برای m و 2015 را برای سال وارد کند، m ، 13 و سال 2014 در فرمول استفاده خواهد شد.)

1. (*مساحت یک چند ضلعی منتظم*) یک چند ضلعی منتظم یک چندضلعی *n* وجهیست که در آن تمام اضلاع دارای طول یکسان و تمام زوایا دارای درجه‌ی یکسانی باشند (یعنی چند ضلعی هم متساوی‌الاضلاع و هم متساوی‌الزوایا است). فرمول محاسبه‌ی مساحت یک چندضلعی منتظم برابر است با:

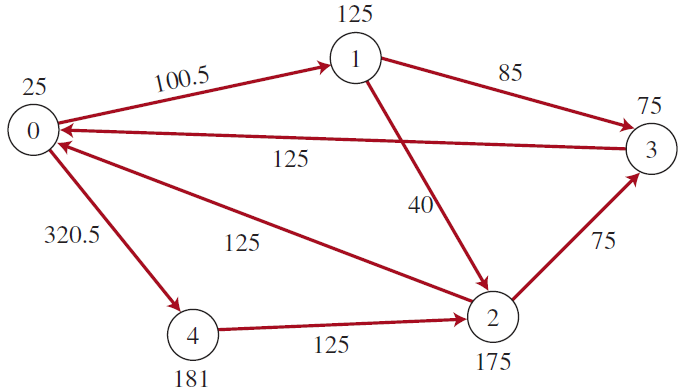
در اینجا s طول یک ضلع است. برنامه‌ای بنویسید که از کاربر بخواهد تعداد اضلاع و طول آنها در یک چند ضلعی منظم را وارد کرده و مساحت آن را نشان دهد.

1. (*مجموع یک سری*) برنامه ای بنویسید که مجموع زیر را محاسبه کند:
2. (*تاریخ و زمان فعلی*) فراخوانی System.currentTimeMillis() زمان سپری شده بر حسب میلی‌ثانیه از نیمه شب 1 ژانویه 1970 را برمی‌گرداند. متدی بنویسید که تاریخ و زمان را نمایش می‌دهد. متد خود را برای چند مقدار دلخواه در یک برنامه‌ی آزمایشی فراخوانی کنید.
3. *(افراز یک لیست)* متد زیر را بنویسید که لیست را با استفاده از عنصر اول که *لولا* نامیده می شود افراز می‌کند:

**public static int** partition(**int**[] list)

پس از افراز ، عناصر موجود در لیست مجدداً مرتب می شوند به طوری که همه‌ی عناصر قبل از لولا کمتر یا مساوی با لولا و عناصر بعد از لولا بزرگتر از لولا هستند. متد اندیسی را که لولا در لیست جدید قرار دارد برمی‌گرداند. برای مثال، فرض کنید لیست {5, 2, 9, 3, 6, 8} باشد. بعد از افراز، لیست به {3, 2, 5, 9, 6, 8} تبدیل می شود. متد را به گونه ای پیاده سازی کنید حداکثر list.length مقایسه طول بکشد. برنامه‌ی آزمایشی بنویسید که از کاربر می‌خواهد اندازه‌ و محتویات لیست را وارد کند و لیست را بعد از افراز نمایش دهد.

1. (*سونامی مالی*) بانک ها به یکدیگر قرض می‌دهند. در شرایط سخت اقتصادی، اگر بانکی ورشکست شود، ممکن است نتواند قرض را بازپرداخت کند. کل دارایی‌های یک بانک موجودی فعلی آن به اضافه‌ی قرض‌هایش به بانک‌های دیگر است. نمودار شکل زیر پنج بانک را نشان می‌دهد. موجودی فعلی بانک‌ها به ترتیب 25، 125، 175، 75 و 181 میلیون دلار است. یال جهت دار از گره 1 به گره 2 نشان دهنده‌ی این است که بانک 1، 40 میلیون دلار به بانک 2 قرض می‌دهد.



اگر مجموع دارایی‌های یک بانک کمتر از حد معینی باشد، بانک ناامن است. پولی که آن بانک قرض گرفته نمی‌تواند به قرض‌دهنده بازگردانده شود، و قرض دهنده نمی‌تواند قرض را در کل دارایی‌های خود به حساب آورد. در نتیجه، قرض‌دهنده نیز اگر مجموع دارایی‌هایش زیر حد باشد ممکن است ناامن باشد. برنامه‌ای بنویسید که تمام بانک‌های ناامن را پیدا می‌کند. برنامه‌ی شما ورودی را به شرح زیر می‌خواند. برنامه ابتدا دو عدد صحیح *n* و *limit* را می‌خواند، که در آن *n* نشان‌دهنده‌ی تعداد بانک‌ها و *limit* حداقل کل دارایی برای امن نگه داشتن یک بانک است. برنامه سپس *n* خط را می‌خواند که اطلاعات برای n بانک با شناسه‌های 0 تا *n* - 1 را توصیف می‌کند.

اولین عدد در ردیف موجودی بانک است، عدد دوم تعداد بانک‌هایی که از بانک قرض گرفته‌اند را نشان می‌دهد، و بقیه جفت دو عدد هستند. هر جفت یک قرض‌گیرنده را توصیف می کند. اولین عدد در جفت شناسه‌ی قرض‌گیرنده و دومی مبلغ قرض‌گرفته‌شده است. به عنوان مثال، ورودی برای پنج بانک در شکل بالا به شرح زیر است (توجه داشته باشید که حد 201 است):

5 201

25 2 1 100.5 4 320.5

125 2 2 40 3 85

175 2 0 125 3 75

75 1 0 125

181 1 2 125

کل دارایی های بانک 3 (125 + 75) است که زیر 201 است، بنابراین بانک 3 ناامن است. پس از ناامن شدن بانک 3، کل دارایی‌های بانک 1 به زیر حد ممکن (40 + 125) می‌رسد. بنابراین، بانک 1 نیز ناامن است. خروجی برنامه باید به صورت زیر باشد:

Here are the unsafe banks: 3 1

(راهنمایی: از یک آرایه‌ی دو بعدی مثل b برای نشان دادن قرض‌ها استفاده کنید. *b[i][j]* نشان‌دهنده‌ی قرضیست که بانک *i* به بانک *j* می‌دهد. زمانی که بانک *j* ناامن می‌شود، *b[i][j]* باید روی 0 تنظیم شود.)

1. Comment [↑](#footnote-ref-1)