به نام خدا

درس مبانی برنامه‌سازی

تمرین ۴

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال اول ۰۰ ـ ۹۹

استاد:

رضا فکوری

مهلت ارسال:

۱۲ دی ـ ساعت ۲۳:۵۹:۵۹

مسئول تمرین‌ها:

امیرمهدی نامجو، پرهام صارمی

مسئول تمرین ۴:

صابر ظفرپور

طراحان تمرین ۴:

سایه جارالهی

علیرضا هنرور

محمدامین آریان

مریم‌سادات رضوی

نوید اسلامی

فهرست

نکات قابل توجه

- توجه داشته باشید که در این تمرین مجاز به استفاده از مباحث بعد از تابع بازگشتی مانند آرایه، استرینگ، پوینتر و ... نیستید.
- تمامی سوالات را باید با استفاده از توابع بازگشتی حل کنید. در غیراینصورت نمره سوال را به طور کامل از دست می دهید.
- سوالات و ابهامات خود درباره‌ی تمرین را در کوئرا مطرح کنید.

سوالات

سوال ۱. Hello Friend

”Hello Friend.
Friend,that’s lame. Maybe I should give you a name.”

الیوت جوانی گوشه گیر است که دستی بر برنامه نویسی و امنیت شبکه دارد. او به عنوان مهندس شبکه در شرکتی به اسم آل سیف در نیویورک مشغول به کار می باشد؛ اما داستان او به اینجا ختم نمیشود. او تفکرات عجیبی در ذهن خود دارد اما از آنجایی که به کسی اعتماد ندارد، در ذهنش با شما حرف میزند. او می گوید گروهی از افراد قدرتمند وجود دارند که این افراد به طور مخفی دنیا را اداره میکنند؛ گروهی که هیچکس از آنها اطلاعی ندارد؛ گروهی که هرکاری بخواهند میکنند و نیاز به اجازه هیچکس ندارند! همچنین او فکر میکند آنها او را دنبال میکنند و میخواهد قبل از اینکه دستشان به او برسد، خودش اولین ضربه را به آنها بزند. او قصد دارد اینکار را از شرکتی که فکر میکند پوششی برای این گروه است، آغاز کند. او میخواهد برنامهای بنویسد که اطلاعات تعدادی از پایگاههای داده این شرکت(که Corp E نام دارد) را رمزگذاری کند تا آن اطلاعات غیر قابل استفاده شود و آسیبی جدی به آنها وارد کند و خودی نشان بدهد.
الیوت به علت تبحر زیاد در برنامه نویسی، میتواند برنامه تخریب یک پایگاه داده را در یک خط کد بنویسد! همچنین واضح است که تخریب صفر پایگاه داده نیاز به هیچ کدی ندارد. اما برای تعداد بیشتر از یک پایگاه داده، رابطه ای برای تعداد خط کد مورد نیاز برای نوشتن برنامه نیاز است که به صورت زیر نوشته میشود:

$$T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + n^2 \quad | \quad n < 500$$

$$T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + 2n^2 \quad | \quad n \geq 500$$

$$T(0) = 0$$

$$T(1) = 1$$

ورودی

ورودی تنها شامل یک عدد n است که برابر تعداد پایگاه دادههایی است که قرار است رمزگذاری شوند.

$$0 \leq n \leq 3000$$

خروجی

خروجی برنامه شما یک عدد است که تعداد خط کد مورد نیاز برنامه است.

مثال

ورودی نمونه ۱

```
1

```

خروجی نمونه ۱

```
1

```

ورودی نمونه ۲

```
600

```

خروجی نمونه ۲

```
1124967

```

توجه کنید که چون عدد ورودی از ۵۰۰ بزرگ‌تر است، کل $T(n)$ ها با تابع دوم محاسبه می‌شوند.

سوال ۲. همه چی آرومه

طبق برنامه، عملیات رمزگذاری پایگاه داده های شرکت با موفقیت انجام شد. بخش زیادی از اطلاعات مورد حمله واقع شده رمزگذاری شد و از بین رفت؛ اما در رسانه ها هیچ خبری مبنی بر حمله سایبری به شرکت داده نشده است. شاید نمیخواهند هیچکس از شکستی که خوردند خبردار شود و ضعف نشان بدهند. الیوت این افکار را با خود مرور میکند. چندروزی از انجامگرفتن حمله به گذشته و به جز عدم پوشش خبر حمله توسط رسانه‌ها مشکلی وجود ندارد؛ تا اینکه او یک تماس عجیب از یک شماره ناشناس دریافت میکند. او تلفن را برمیدارد و این جملات را با صدای ضبط شده میشنود: "الیوت آلدرسون؛ ۲۶ ساله؛ مهندس ارشد شبکه در شرکت امنیت سایبری آلسیف؛ ساکن در نیویورک، بروکلین، خیابان اسپرینگ پلاک ۷۶. عامل حمله به پایگاه داده شرکت. بله میدانیم کار شماست. هیچکاری نکنید و منتظر باشید".

لعنت.مثل اینکه درگیرشدن با قدرتمندترین انسانهای روی زمین آنقدرها هم آسان نیست. اما منتظر نشستن هم کار درستی به نظر نمی آید. برای همین الیوت دست به کار میشود. او تصمیم میگیرد با هک کردن برجهای مخابراتی که احتمال تماس از طریق آنها وجود دارد، آدرس دقیق مکانی تماس از آنجا گرفته شده را به دست آورد و سری به آنجا بزند تا بلکه بتواند دوباره نسبت به آنها برتری پیدا کند. او ۶ برج مخابراتی که بیشترین احتمال برقراری تماس از آنها وجود دارد را هک کرده و اطلاعات a تا f را از آنها به دست می آورد. برای به دست آوردن مختصات دقیق محل تماس، نیاز است تا این اطلاعات به دست آمده با روش صحیحی پردازش شوند. به این صورت که از a تا f به عنوان ضریب یک معادله درجه ۵ استفاده میکند و همچنین تعیین میکند که در این قسمت از شهر(که به صورت دو عدد برای مشخص کردن بازه جواب داده میشود) به دنبال مختصات مورد نظر بگردد و در صورت پیدا شدن، آن را چاپ کند.

$$ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0$$

روش پیدا کردن جواب معادله در بازه مورد نظر به این صورت است:ابتدا مقدار تابع در دو سر بازه مورد نظر را پیدا میکند.در صورتی که مقدار تابع در دو سر بازه هم علامت بودند،عبارت NOT POSSIBLE! چاپ میشود در غیر این صورت مقدار تابع در نقطه وسط بازه را پیدا میکند و علامت آن را با مقدار تابع در دوسر بازه مقایسه میکند و سپس این عمل را برای نیم بازه ای تکرار میکند که مقدار تابع در دوسر آن هم علامت نیستند.

ورودی

در خط اول ورودی، هر یک از ضرایب اعشاری a تا f به ترتیب و با یک فاصله از هم آمده اند. در خط دوم ورودی دو عدد اعشاری m و n که دو سر بازه ای هستند که جواب در آنها پیدا میشود، داده شده است.

$$0 \leq n \leq 3000$$

خروجی

خروجی برنامه تنها یک خط است، که در صورتی که مقادیر دو سر بازه اولیه هم علامت باشند عبارت

$$NOTPOSSIBLE!$$

چاپ میشود و در غیر این صورت، پاسخ معادله به صورت عدد اعشاری چاپ می شود.

مثال

ورودی نمونه ۱

```
0 0 0 0 2 -22
0 20

```

خروجی نمونه ۱

```
11.000000

```

معادله به صورت $2x = 22$ است و در نتیجه خروجی 11 میشود.

ورودی نمونه ۲

```
0 0 0 1 -5 6
0 2.5

```

خروجی نمونه ۲

```
2.000000

```

معادله به صورت $x^2 - 5x + 6 = 0$ است و دربازه ۰ تا ۵.۲ تنها جواب برابر $x = 2$ است. ابتدا مقدار تابع در دو سر بازه بررسی می شود. در $x = 0$ مقدار تابع برابر 6 و در $x = 2.5$ مقدار تابع برابر با -0.25 است. با توجه به اینکه مقدار تابع در دو سر بازه هم علامت نیستند، میتوان جواب را در این بازه پیدا کرد و جواب به طور بازگشتی پیدا می شود.

ورودی نمونه ۳

```
0 0 0 1 6 10
-10 10

```

خروجی نمونه ۳

```
NOT POSSIBLE!

```

مقدار تابع در $x = 10$ برابر با 170 و در $x = -10$ برابر با 50 است. با توجه به اینکه این مقادیر هم علامتند، نمیتوان جوابی برای معادله ارائه داد.

سوال ۳. وقت حرف زدنه

الیوت که با موفقیت مختصات محل تماس را به دست آورده، تصمیم می‌گیرد سری به آن‌جا بزند. برای همین با تکه‌کاغذی که روی آن آدرس محل مورد نظر را نوشته از آپارتمانش در خیابان اسپرینگ خارج می‌شود، اما به محض اینکه پایش را از در بیرون می‌گذارد، کیسه‌ای پارچه روی سرش کشیده می‌شود و قبل از اینکه دست بجنبد، چند نفر او را سوار چیزی که به نظر می‌رسد یک ون باشد(با توجه به صداهایی که می‌شنود)، می‌کنند و فریادهای الیوت کمکی به او نمی‌کنند.

وقتی ون به مقصد می‌رسد، او را پیاده می‌کنند و پس از طی مسافت اندکی با چشمان بسته، کیسه را از سر او برمی‌دارند و الیوت، خود را در راهروی نسبتا باریک که در انتهای آن یک در شیشه‌ای است، می‌یابد. او جلو می‌رود و در را باز می‌کند و وارد اتاقی با دیوارهای سفید و نور نسبتا زیاد می‌شود که در وسط آن، دو صندلی و در میان آن‌ها، یک میز و روی این میز، یک کامپیوتر به‌نظر قدیمی قرار دارد. او مدت‌زمان کوتاهی سرگردان در اتاق می‌چرخد تا اینکه صدایی توجهش را جلب می‌کند: "خوش آمدید آقای آلدرسون"! از دیگر در اتاق، شخصی وارد می‌شود: "لطفا بنشینید". الیوت بلافاصله می‌پرسد: "برای چی من رو آوردید اینجا؟". آن شخص پشت میز می‌نشیند، کاغذی را از جیبش بیرون می‌آورد که همان کاغذی است که الیوت هنگام خارج شدن از آپارتمانش همراه داشت. آن شخص* به کاغذ نگاه می‌کند و می‌گوید: "خیلی برای پیدا کردن ما مشتاق بودید؛ حالا به هدفتان رسیدید. حالا قرار است با هم حرف بزنیم. می‌توانید بایستید اما از آن‌جایی که احتمالا صحبت طولانی‌ای با هم خواهیم داشت، من ترجیح می‌دهم نشسته با شما صحبت کنم". الیوت نیز روی صندلی می‌نشیند. آن شخص شروع به صحبت می‌کند: "از بین بردن پایگاه‌داده‌های شرکت E Corp عملیات تحسین‌برانگیزی بود، اما شما واقعا به درست‌بودن کارهایتان یقین دارید؟ فکر می‌کنید ما آدم‌های پلیدی هستیم و شما قرار است جلوی مارا بگیرید؟". الیوت یکی عقب افتاده است. او نیاز دارد جواب محکمی به آن شخص بدهد تا در نهایت بتواند در بحث پیروز شود و **کم نیاورد**. حال شما باید **بقیه** مکالمه بین الیوت و آن شخص مرموز را دریافت کنید و طول مکالمه و پیروزشدن یا شکست خوردن الیوت در بحث را مشخص کنید؛ اما این مکالمه شروطی دارد:

۱. ابتدا الیوت باید تمام حرف‌های آن شخص را گوش کند و سپس شروع به پاسخ دادن کند.

۲. چون آن شخص تمام حرف‌هایش را قبل از الیوت گفته است، هرجا که الیوت از لحاظ تعداد جملات تاثیرگذار به او رسید، بحث تمام‌شده محسوب شده و الیوت برنده می‌شود.

۳. جملات دندان‌شکن شخص مرموز را با a و جملات دندان‌شکن الیوت را با b مشخص می‌شود. همچنین الیوت برای پایان مکالمه از حرف c استفاده می‌کند.

*توجه کنید که «آن شخص» از کلماتی مانند `while` و `for` متنفر است و در صورت مشاهده آن‌ها به کد نمره **صفر** می‌دهد!

ورودی

در خط اول مکالمه به صورت دنباله‌ای از a و b و c داده می‌شود.

خروجی

خروجی در یک خط به فرمت زیر چاپ می‌شود که در آن num طول مکالمه و msg در صورت پیروز شدن الیوت YES و در صورت شکست خوردن او NO می‌باشد. فرمت :

```
num. msg

```

مثال

ورودی نمونه ۱

```
aabbbc

```

خروجی نمونه ۱

```
6. YES

```

دنباله چک شده $aaabbb$ هست که طولش ۶ هست. به سومین b که رسید دنباله خوب پیدا شد پس بقیه دنباله رو نادیده گرفت!

ورودی نمونه ۲

```
aabbc

```

خروجی نمونه ۲

```
6. NO

```

از اونجایی که تو دنبالمون ۳تا a داریم، باید ۳تا b می‌دیدیم ولی به c برخوردیم و کار خراب شد! طول دنباله چک شده هم ۶ بود.

سوال ۴. حرف زدن بسه

بعد از یک مکالمه طولانی و طاقت‌فرسا، شخصی که برای قانع کردن الیوت آمده بود، به این نتیجه می‌رسد که عقیده او تغییرپذیر نیست و نمی‌توان از او استفاده کرد. برای همین تصمیم می‌گیرد از راه دیگری وارد شود. او به الیوت می‌گوید که با استفاده از این کامپیوتر قدیمی که مستقیماً به پایگاه‌داده‌های شرکت متصل است، کلید مربوط به رمزگشایی اطلاعات را وارد کند. مشخصاً الیوت زیر بار نمی‌رود و می‌پرسد که چرا باید این‌کار را انجام دهد؟ آن شخص هم او را تهدید می‌کند که اگر این‌کار را انجام ندهد، حادثه ناگواری در آپارتمانش و شرکت آل‌سیف (که الیوت در آن کار می‌کند) رخ خواهد داد. الیوت که می‌داند با چه گروه خطرناکی مواجه است، چاره‌ای جز انجام این‌کار ندارد. او پشت کامپیوتر می‌نشیند و شروع به ساخت کلید می‌کند.

کلید مورد نظر یک برنامه است که n بار عدد 7 را با طول و عرض های $2(n-i)$ که در آن i از 0 تا $n-1$ است، با " $*$ " رسم می‌کند. الیوت موظف است که برای این‌کار از تابع بازگشتی استفاده کند.

ورودی

ورودی شامل عدد صحیح n است.

$$0 \leq n \leq 100$$

خروجی

خروجی متشکل از 7 های ستاره ای زیر هم است که با طول و عرض $2n$ شروع شده و به مرور کوچک می‌شوند تا به 7 با طول و عرض 2 کارکتر برسیم. برای فهم بهتر سوال به نمونه های ورودی و خروجی توجه کنید.

مثال

ورودی نمونه ۱

```
3

```

خروجی نمونه ۱

```
******
     *
    *
   *
  *
*
****
   *
  *
*
**
*

```

ورودی نمونه ۲

```
5

```

خروجی نمونه ۲

```
**********
         *
        *
       *
      *
     *
    *
   *
  *
*
********
       *
      *
     *
    *
   *
  *
*
******
     *
    *
   *
  *
*
****
   *
  *
*
**
*

```

سوال ۵. گربه شرودینگر

آلارم ساعت کنار تخت به صدا در می‌آید. دستی بی هوا به آن برخورد می‌کند و آلارم خاموش می‌شود. ساعت 07:00 صبح را نشان می دهد. الیوت از خواب بیدار می‌شود!

مانند برنامه‌ی روتین زندگی‌اش، برای سر کار رفتن آماده می‌شود. پس از دوش گرفتن روزانه‌اش، هودی مشکی ساده‌ی همیشگی‌اش را به تن می‌کند تا به شرکت برود و صبحانه‌اش را که یک لیوان قهوه و مقداری کیک است،آنجا بخورد؛ اما او احساس عجیبی دارد. شما ندارید؟ به نظر می‌رسد حافظه‌اش مختل شده است. او هیچی از چند روز اخیرش یادش نمی‌آید،انگار به جای از بین بردن اطلاعات E-Corp، اطلاعات خود را از بین برده است! «اطلاعات E-Corp؟ لعنت. قرار بود پایگاه داده‌شون رو از بین ببرم... صبر کن! این کار رو کردم؛ نه؟»
الیوت سریعا این را روی کاغذ می‌نویسد. او مطمئن می‌شود که مشکلی وجود دارد و تلاش می‌کند هر چه را به خاطر می‌آورد روی کاغذ بنویسد. بعد از مدت زمانی، اوو می‌بیند تعدادی خاطره که ترتیب مشخصی ندارند،روی کاغذ نوشته شده است. برای فهمیدن این که چه اتفاقی افتاده، الیوت نیاز دارد تا آن خاطره‌ها را به یک ترتیبی مرتب کند؛ به همین دلیل، کنار هر خاطره یک شماره‌ی تصادفی مثبت می‌نویسد و این شماره‌ها را به صورت دنباله‌ای از چپ به راست در جایی دیگر می‌نویسد. سپس برای هر شماره، راست‌ترین شماره‌ی سمت چپش که از آن شماره بزگ‌تر نیست را در دنباله‌ای دیگر می‌نویسد تا ترتیب جدیدی از شماره‌ها را به دست آورد تا با این ترتیب جدید بفهمد چه اتفاقی افتاده. اما واقعا چه اتفاقی افتاده؟

ورودی

ورودی دنباله‌ای از شماره‌های ذکر شده در بالا است و در انتهای دنباله، عدد 0 برای اتمام اعداد ورودی داده می‌شود. طول این دنباله در واقع n است.

$$1 \leq n \leq 10^5$$

خروجی

برای هر شماره، مقدار راست‌ترین شماره‌ی چپ آن را که ازش بزرگ‌تر نیست را چاپ کنید. همچنین اگر برای یک شماره چنین مقداری موجود نبود، مقدار -1 را خروجی دهید.

مثال

ورودی نمونه ۱

```
1 2 1 0

```

خروجی نمونه ۱

```
-1
1
1

```

شماره‌ی اول، چون شماره‌ی چپ‌تر از خودش ندارد، جواب -1 دارد. سایر شماره‌ها نیز طبق تعریف جواب، باید جواب 1 داشته باشند. چون تنها شماره‌ی کمتر مساوی آن‌ها در سمت آن‌ها، اولین شماره است.

ورودی نمونه ۲

```
1 2 3 1 4 5 7 2 8 3 0

```

خروجی نمونه ۲

```
-1
1
2
1
1
4
5
1
2
2

```

پاسخ‌ها

پاسخ ۱. Hello Friend

نمونه پاسخ مورد قبول:

```c
#include <stdio.h>
int t(int);
int flag = 0;

int main() {
    int n;
    scanf("%d", &n);
    if(n >= 500) {
        flag = 1;
    }
    printf("%d\n", t(n));
    return 0;
}

int t(int n) {
    if(n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }
    int result;
    if(flag == 0) {
        result = (t(n / 2) + t(n / 3) + n * n);
    } else {
        result = (t(n / 2) + t(n / 3) + 2 * n * n);
    }
    return result;
}

```

برای حل این سوال ابتدا ورودی n را در تابع $main$ می‌گیریم. سپس از یک متغیر $flag$ استفاده می‌کنیم و اگر n بزرگ‌تر یا مساوی 500 بود، متغیر $flag$ را برابر 1 قرار می‌دهیم. حال تابع بازگشتی را صدا زده و به عنوان ورودی، n را به آن می‌دهیم. در تابع بازگشتی، ابتدا ورودی تابع را بررسی می‌کنیم؛ اگر $n = 0$ بود، تابع مقدار صفر و اگر $n = 1$ بود مقدار یک را برمی‌گرداند. در صورتی که $n > 1$ باشد، متغیر $flag$ بررسی می‌شود؛ اگر

یک بود از تابع اول و اگر صفر بود از تابع دوم ارائه‌شده در صورت سوال استفاده می‌شود. در آخر نیز تابع بازگشتی مقدار متغیر $result$ را برمی‌گرداند.

پاسخ ۲. همه چی آرومه

نمونه پاسخ مورد قبول:

```c
#include <stdio.h>
#include<math.h>
double a,b,c,d,e,f;
double resolve(double x){
    return a*pow(x,5)+b*pow(x,4)+c*pow(x,3)+d*pow(x,2)+e*x+f;
}
double findX(double min,double max){
    if(resolve(min)==0)
        return min;
    if(resolve(max)==0)
        return max;
    else{
        double c = (max+min)/2.0;
        if(resolve(c)==0.0)
            return c;
        else if(resolve(c)*resolve(min)<0){
            return findX(min,c);
        }else{
            return findX(c,max);
        }
    }
}
int main()
{
    double min,max;
    scanf("%lf %lf %lf %lf %lf %lf",&a,&b,&c,&d,&e,&f);
    scanf("%lf %lf",&min,&max);
    if(resolve(min)*resolve(max)>0){
        printf("NOT POSSIBLE!");
        return 0;
    }
    double result =findX(min,max);
    printf("%lf",result);
    return 0;
}

```

تابع $resolve$ مقدار تابع را در نقطه ای که ورودی داده شده است، خروجی می دهد. ابتدا در تابع $main$ بررسی می شود که مقادیر تابع در دو بازه داده شده هم علامت هستند

یا خیر. در صورتی که این دو مقدار هم علامت باشند، عبارت !NOT POSSIBLE چاپ میشود و برنامه پایان می یابد. در غیر اینصورت به طور بازگشتی جواب تابع پیدا می شود. الگوریتم حل مسئله مطابق با روش $bisection$ است. در این روش هربار مقدار تابع در وسط بازه داده شده محاسبه می شود. در صورتی که صفر باشد جواب مسئله پیدا شده است و در غیر اینصورت، نیم بازه ای انتخاب می شود که در دوسر آن مقادیر تابع هم علامت نباشند.

پاسخ ۳. وقت حرف زدنه

نمونه پاسخ مورد قبول:

```c
#include <stdio.h>
#include <stdlib.h>

int count = 2;

void F() {
    char input;
    scanf("%c", &input);
    if (input == 'a') {
        count++;
        F();
    }
    else
        return;
    scanf("%c", &input);
    count++;
    if (input == 'a' || input == 'c') {
        printf("%d. NO\n", count);
        exit(0);
    }
}

int main() {
    F();
    printf("%d. YES\n", count);
    return 0;
}

```

در این سوال ابتدا تعداد a ها را می‌شماریم. این کار را با استفاده از if اولی انجام می‌دهیم. یک $count$ تعریف می‌کنیم تا تعداد کاراکترهای حساب‌شده را نگه داریم. اولین کاراکتری که a نبود، باعث $return$ شدن می‌شود. وقتی $return$ صورت گیرد، با تابع قبلی خود مواجه می‌شود (کد از خط ۱۱ ادامه پیدا می‌کند) که به خط $scan$ دومی می‌رسد. اگر کاراکتر بعدی a یا c بود یعنی بعد از b با a یا c مواجه شده‌ایم که به معنی خراب شدن گفتگو است. بدین ترتیب چاپ را انجام داده و برنامه را تمام می‌کنیم. اگر غیر از این بود، تابع تمام می‌شود و به $main$ برمی‌گردیم.

پاسخ ۴. حرف زدن بسه

نمونه پاسخ مورد قبول (کد آقای کیارش کیانیان):

```c
#include <stdio.h>

int star(int n,int i) {
  if(i>n)
    return 0 ;
  else{
    printSpace(n - i,0);
    printf("*\n");
    star(n,i+1);
  }
}

int printStar(int length, int i) {
  if(i>=length)
    return 0;
  else{
    printf("*");
    printStar(length, i+1);
  }
}

int printSpace(int length, int i) {
  if(i>=length)
    return 0;
  else{
    printf(" ");
    printSpace(length,i+1);
  }
}

int create(int n){
  if (n==0)
    return 0;
  else {
    printStar(2*n, 0);
    printf("\n");
    star(2 * n,2);
    create(n-1);
  }
}
```

```
41
42 int main(){
43   int n;
44   scanf("%d", &n);
45   create(n);
46   return 0;
47 }
48
49
```

توجه کنید که برای حل این سوال راه‌های بازگشتی و غیر بازگشتی دیگری نیز وجود دارد و پاسخ آورده شده یک نمونه است.

پاسخ ۵. گربه‌ی شرودینگر

نمونه پاسخ مورد قبول:

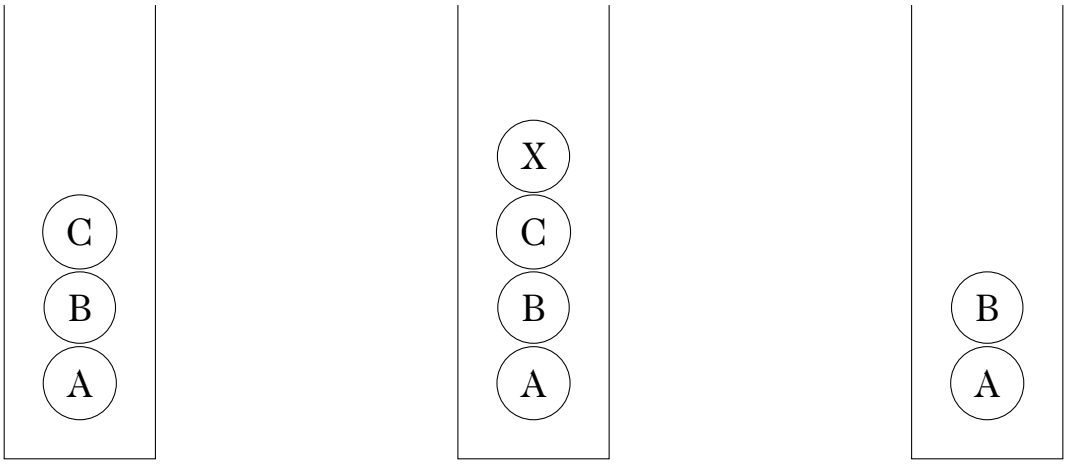
```c
#include <stdio.h>

int solve(int prevVal) {
  int currentValue;
  scanf("%d", &currentValue);
  if (currentValue == 0) {
    return currentValue;
  }
  if (prevVal <= currentValue) {
    printf("%d\n", prevVal);
    while (1) {
      int tmp = solve(currentValue);
      if (tmp == 0) {
        return tmp;
      }
      if (prevVal <= tmp) {
        printf("%d\n", prevVal);
        currentValue = tmp;
      }
      else {
        return tmp;
      }
    }
  }
  else {
    return currentValue;
  }
}

int main() {
  solve(-1);
}

```

نمونه‌ی کد کوتاه‌تر، ولی با همان راه حل:

```
#include <stdio.h>

int solve(int lastLink) {
  int currentLink;
  scanf("%d", &currentLink);
  if (currentLink == 0) {
    return 0;
  }
    while (lastLink <= currentLink && currentLink != 0) {
        printf("%d\n", lastLink);
        currentLink = solve(currentLink);
    }
    if (lastLink > currentLink) {
        return currentLink;
  }
}

int main() {
    solve(-1);
}

```

همانطور که ذکر شد، این دو کد در واقع مربوط به یک راه حل واحد هستند، که در ادامه راه حل را توضیح می‌دهیم و کد دوم را تشریح می‌کنیم.
برای توضیح راه حل، اول با داده‌ساختار استک آشنا می‌شویم. داده‌ساختار استک، یک داده‌ساختار ساده است مطابق با شماتیک فوق، که دارای سه عمل زیر می‌باشد:

- push(x): مقدار x را به سر ساستک اضافه می‌کنیم.
- pop(): عنصر سر استک را پاک می‌کند.
- top(): مقدار سر استک را به ما می‌دهد می‌کند.



در شکل سمت چپ، یک استک با سه عنصر نمایش داده شده است. اگر عمل push(x) را روی آن اجرا کنیم به استک وسطی می‌رسیم. همچنین اگر عمل pop() را روی استک چپ انجام دهیم، به استک راست خواهیم رسید.

حال با استفاده از این داده‌ساختار، الگوریتم زیر را برای سوال خود ارائه می‌دهیم:

۱. یک ااستک خالی به نام S تعریف می‌کنیم.

۲. به ازای هر عنصر جدید x که مشاهده می‌کنیم، تا وقتی که استک ما خالی نبود و عنصر سر استک، بزرگ‌تر از x بود، عمل pop را انجام می‌دهیم.

۳. سپس، می‌فهمیم که یا S خالی شده است، یا عنصر سر آن از x بزرگ‌تر نیست. در حالت اول، خروجی مربوط به x، ۱ـ است. در حالت دوم، خروجی مربوط به x همان عنصر سر استک است.

۴. حال x را به S push می‌کنیم.

۵. مراحال ۲ تا ۴ را تا زمانی که صفر ورودی نگرفته‌ایم اجرا می‌کنیم.

حال ثابت می‌کنیم که این الگوریتم جواب صحیح مربوط به مسئله را چاپ می‌کند. طبق روند الگوریتم، می‌دانیم که عناصر به ترتیب، از چپ به راست به S اضافه می‌شوند. به عبارتی، اگر a سمت چپ b باشد، طبق گام ۴ام، می‌دانیم که a زودتر از b وارد S شده است. حال با استفاده از این نکته و به کمک استقرا، درستی الگوریتم را ثابت می‌کنیم.

حکم استقرا: الگوریتم ما، در هر مرحله از جواب دادن به عناصر، جواب صحیح را چاپ می‌کند. همچنین بعد از هر مرحله، محتویات S به صورت یک دنباله‌ی نزولی (از بالا به پایین) خواهد بود که تمام عناصری که در آن نیستند، نمی‌توانند کاندیدی برای جواب یک عدد در آینده باشند.

- پایه‌ی استقرا: اولی عنصر که به S اضافه می‌شود، چون قبل از آن S خالی بوده است، جواب ۱ـ را برای آن خواهیم داشت. که با توجه به تعریف سوال، جواب صحیح مربوط به آن است. همچنین خود آن عنصر هم تا اینجا، تنها کاندید موجود برای جواب بودن بقیه‌ی عناصر است.
- گام استقرا: فرض کنید در مرحله‌ای قرار داریم که عنصر x را ورودی گرفته‌ایم. می‌دانیم، طبق فرض استقرا، که S دارای شرایط حکم ما است و هیچ عنصری که تا این مرحله دیده شده است و در S نیست، نمی‌توانند جواب مربوط به x باشند. حال الگوریتم ما، تا جایی که عناصر سر S از x بزرگتر بودند، آن‌ها را پاک می‌کند. در آخر، به یک عنصر می‌رسیم مثل y که $y \leq x$. طبق نکته‌ای که در ابتدا گفتیم، می‌فهمیم که y از سایر عناصر حاضر در S راست‌تر است. همچنین چون راست‌ترین عنصری است که کاندید مناسبی برای جواب x بودن است، می‌فهمیم که جواب x همان y است! اگر هم S خالی شد، در واقع هیچ کدام از کاندیدها نمی‌توانستند جواب مناسبی برای x باشند و در واقع جواب x، همان ۱ـ خواهد بود.
حال باید ثابت کنیم که S بعد از انجام اعمال بالا، شرایط حکم استقرا را داراست. اول از نزولی بودن شروع می‌کنیم. می‌دانیم که بعد از این مرحله، یا فقط x در S است، یا یک سری عناصر که y بالایی‌ترین آن‌ها بود، زیر x هستند. در حالت اول، S به وضوح نزولی است. در حالت دوم می‌دانستیم که تمام عناصر، بدون در نظر گرفتن x، نزولی خواهند بود. همچنین داشتیم که $y \leq x$، پس طبق خاصیت تعدی می‌فهمیم که x از همه‌ی عناصر دیگر بزرگ‌تر یا مساوی است. پس می‌فهمیم که خاصیت نزولی بودن برقرار است. حال سراغ خاصیت داشتن تمام کاندیدها می‌رویم. هر عنصری مثل z که از S پاک شد، می‌دانیم که $z > x$ بود. همچنین می‌دانیم که z سمت چپ x بود! پس می‌فهمیم که هیچ‌کدام از آن zها نمی‌توانست جواب عنصری بعد از x باشد، چون x راست‌تر از آن‌ها است و مقدار کمتری از آن‌ها دارد. پس برای هر عددی سمت راست x، خود x قطعا کاندید بهتری از هرکدام از zها برای جواب بودن خواهد بود.

پس می‌توان تمام zها را حذف کرد. سایر عناصر هم چون دستشان نزدیم، تغییری در کاندید بودن آن‌ها برای عناصر بعدی رخ نخواهد داد.

پس حکم را ثابت کردیم و مسئله با استفاده از استک حل شد! حال، با استفاده از تابع بازگشتی این راه حل با استک را شبیه سازی می‌کنیم.
هر تابع بازگشتی، در هر مرحله، شکلی معادل با یک زنجیر دارد. به عبارتی، هر حلقه‌ی زنجیر سمت چپ، حلقه‌ی زنجیر سمت راستش را ساخته است: (A)(B)(C)(D)(E)

برای مثال، در شکل بالا، A در واقع B را ساخته است، C B را و بقیه هم به همین شکل.
حال به این زنجیر، به این صورت نگاه می‌کنیم که راست‌ترین حلقه، در واقع سر استک ما باشد و در عین حال، چپ‌ترین حلقه پایین‌ترین عنصر استک باشد. پس با استفاده از این، استک را شبیه سازی می‌کنیم. هر موقع خواستیم push کنیم، در آخرین حلقه تابع را صدا می‌زنیم و هر موقع خواستیم pop کنیم، return می‌کنیم. همچنین، در هر حلقه اگر یک عدد ذخیره کنیم و هر حلقه عدد خودش را به عنوان ورودی به تابع بعدی که در واقع حلقه‌ی بعدی است بدهد، می‌توانیم به راحتی دستور top را هم داشته باشیم! پس با استفاده از این نکات، کد را می‌زنیم.
در ابتدا، ما solve(۱ -) را صدا زده بودیم. این به این معنی است که در پشته‌ی خود اول عدد ۱ـ را به استک مجازی خود اضافه کرده‌ایم. با استفاده از این حلقه از زنجیر خود، می‌توانیم تشخیص دهیم که استک ما خالی هست یا نه. حال به کد داخل هر حلقه از زنجیر می‌پردازیم. در هر کدام از آن حلقه‌ها، یک عدد $currentLink$ ورودی می‌گیریم. این متغیر در واقع مقدار ذخیره شده در هر حلقه از زنجیر را نشان می‌دهد. اگر آن مقدار صفر بود، به انتهای ورودی رسیده‌ایم و باید برنامه را تمام کنیم. در غیر این صورت، وایل نوشته شده را اجرا می‌کنیم. در واقع، اگر مقدار حلقه‌ی قبلی کمتر مساوی با مقدار فعلی بود، جواب مقدار فعلی را یافته‌ایم. پس یک حلقه‌ی جدید برای ادامه‌ی کار می‌سازیم که در واقع $currentLink$ را push کرده‌ایم. حال، اگر تابع solve بعد از مدتی return کند، در واقع دو اتفاق ممکن است افتاده باشد:

- در حال pop کردن از استک هستیم و مقدار آخرین عنصر ورودی گرفته شده را return می‌کنیم.
- صفر ورودی گرفته‌ایم و باید خارج شویم.

در حالت اول، طبق الگوریتم استک، آنقدر return کرده‌ایم که به یک عنصر کمتر یا برابر برسیم. همچنین شرط return کردن ما هم این بود که مقدار قبلی بزرگ‌تر از مقدار فعلی ما باشد. پس در هر حلقه از زنیجر ما که $currentLink$ دوباره مقدار دهی شود، یعنی مقدار قبلی آن بزرگ بوده و باید حذف می‌شد. حال، وایل دوباره اجرا می‌شود و اگر بیشتر مساوی قبلی بودیم، جواب را پیدا کرده‌ایم و دوباره حلقه‌ی جدید زنجیر را می‌سازیم. در غیر این صورت و یا در صورتی که صفر بود، آن مقدار را return می‌کنیم تا به pop کردن ادامه دهیم. توجه کنید که اگر زنجیر ما خالی شود و به حلقه‌ی اول آن که ۱ ـ داشت برسیم، دو حالت داریم. یا مقدار صفر را گرفته‌ایم، که return خواهیم کرد، یا مقداری ناصفر داشتیم، که چون گفته بودیم تمام اعداد مثبت هستند و اگر کسی جواب نداشت ۱ ـ چاپ کنید، همان printf نوشته شده کار مار را راه می‌اندازد. پس می‌فهمیم که کد نوشته شده به صورت صحیح عمل می‌کند و سوال را با استفاده از راه استک، حل می‌کند.