بسمه‌تعالی

پاسخ تمرین سری پنجم

فهرست:

- عموشکسپیر
- ریسمان های پوسیده
- تابع گاما
- Selection-Sort

کلیه برنامه های شما توسط کمپایلر gcc ورژن ۴.۰ و در محیط شلِ لینوکس کمپایل و تست خواهند شد. علی‌القاعده نباید در مثال های ساده ای همچون این تمرین مشکلی داشته باشید. اما تلاش کنید تا در محیط های استاندارد کد بزنید تا در آینده (و به خصوص پروژه هایتان) دچار مشکلی نشوید.
در صورتی که درباره جواب هایی که در این حل تمرین ارائه شده است مشکلی داشتید یا جواب های بهتر و جالب‌تری را بلدید می توانید در صفحه گروه درس به آدرس groups-beta.google.com/group/ce153c یا صفحه discussion سایت درس بیان کنید.

۰ ..... عموشکسپیر

To Be or Not to Be, Calculate the Result!!!

در برنامه‌ی صورت سوال ۲ ایراد وجود داشت، در زبان C بین تقسیم صحیح و اعشاری تفاوت وجود دارد. بنابراین هنگامی که دو عدد صحیح را بر هم تقسیم می کنیم حاصل همچنان به صورت عدد صحیح برگردانده می‌شود و قسمت اعشاری دور ریخته می‌شود. برای گرفتن نتیجه اعشاری می بایست به صورت زیر عمل کنیم (حداقل یکی از عملوند ها می بایست اعشاری باشد.) راه ممکن دیگر استفاده از دو متغیر است تا اعداد را در آنها ذخیره کرده و بعد متغیر ها را بر هم تقسیم کنیم.

result = 1.0 / 3.0 یا result = 1 / 3.0

به علاوه در دستور printf هم مشکل وجود دارد. به جای %f از %o استفاده شده است که باعث می شود خروجی برنامه در مبنای ۸ نوشته شود.
برنامه صحیح:

```
#include <stdio.h>
int main(){
     float result;                        /* Result of the divide */
     result = 1/3.0;                      /* Assign result something */
     printf("Result is %f \n" , result ) ;
     return (0);
}
```

به علاوه بهتر است در سوالاتی مثل این سوال به موارد زیر هم دقت کنید:

- C یک زبان case-sensitive می باشد. یعنی نسبت به بزرگ بودن یا کوچک بودن حروف حساس است. برای همین عبارتی مثل INT main() غلط محسوب می شود.
- امروزه خروجی تابع main در زبان استاندارد C می بایست حتماً int باشد.
- به شکل دستور return دقت کنید. این عبارت می تواند بدون پرانتز هم بیاید.

ب) این سوال مورد علاقه من است! هنگامی که این سوال را به بسیاری از افراد غیرسمینیست! بیان می کنیم اکثراً در ابتدا گیج می شوند.
پاسخ :

```
int x = 0x2B || !(0x2B) /* to be ( 2B ) or ( || ) not (!) to be (2B) */
```

حاصل این عبارت 0xFF می باشد اما اگر آنرا در خروجی چاپ نمایید ۱- خواهید گرفت.(چرا؟ (راهنمایی: زبان C برای اعداد منفی از مکانیزم 2' Complement استفاده می کند.))

ج) این سوال که دیگر نیاز به توضیح ندارد. (فرض کنید $x > y$)

```
int gcd( int x, int y ){
  int i ;
  for ( i = y ; i > 0 ; i--){
    if ( x%i == 0 && y%i == 0) return i;
  }
}
```

به علاوه صورت بازگشتی عبارت فوق به شکل زیر است:

```
int gcd( int x, int y ){
  if ( y==0) return x ;
  else return gcd(y , x%y );
}
```

به نظرتان کدام ساده‌تر است؟ استفاده از توابعی که خودشان را فرا می‌خوانند در بسیاری از مسائلی که یک روند یکسان درباره قسمت های مختلف مسئله صادق است کاراست. به علاوه توجه کنید که هر الگوریتم بازگشتی را می‌توان به گونه غیر بازگشتی خودش تبدیل کرد.
برای تمرین بیشتر بهتر است مسئله زیر را یک بار با کمک روش بازگشتی و یکبار به صورت غیر بازگشتی حل کنید:
مسئله کوله‌پشتی: صفر-یک: یک سارق مسلح نیمه‌شب از یک طلافروشی سرقت می کند. این سارق n بسته طلای مختلف که وزن بسته iاُم برابر m_i و ارزشش برابر با p_i می باشد را برداشته است. اما از طرفی نمی تواند همه آنها را با خودش ببرد چرا که کوله‌پشتی او حداکثر می تواند وزن S را تحمل کند.(طفلکی!).
برنامه ای بنویسید که بتواند زیرمجموعه از از m_i ها با بیشترین ارزش تولید کند به شرط آنکه مجموع m_i ها از S بیشتر نشود.

صورت ریاضی وار و ساده‌شده مسئله فوق به این شکل است که مجموعه A شامل n عدد طبیعی داده شده است. آیا زیرمجموعه ای از A وجود دارد که جمع اعضای آن برابر S شود یا خیر؟
ایده حل بازگشتی این سوال به این شکل است که عضو آخر مجموعه یا در زیرمجموعه مورد نظر ما هست یا نیست. و این کار را به صورت بازگشتی برای بقیه مجموعه تکرار کنیم. بدین صورت که اگر عضو انتهایی درون زیرمجموعه نبود قدم بازگشت را برای $A[first], A[last-۱], n$ تکرار کنیم و در صورتی که در مجموعه بود آنرا برای $A[first], A[last-۱], n-A[last]$.

۱ ..... ریسمان های پوسیده

این سوال برخلاف ظاهرش راه‌حل بسیارساده ای دارد.
ابتدا آرایه شامل نیروی تحمل طناب‌ها ($ropes$) را به صورت نزولی مرتب می‌کنیم. سپس مقدار نیروی ماکزیمم (max) را از روی این آرایه می سازیم. به این شکل که در هر مرحله چک می‌کنیم که آیا طناب بعدی (که مطمئناً ضعیف‌تر است، زیرا آرایه نزولی مرتب شده است.) می تواند (نیروی ماکزیمم × تعداد طناب ها) را تحمل کند یا خیر؟ اگر می توانست آنرا هم به طناب هایمان اضافه می کنیم و این کار را ادامه می‌دهیم. در غیر این‌صورت خارج می شویم.(بدیهی است که دیگر نیازی به چک کردن باقی طناب ها نیست. چرا که همه ضعیف‌ترند.)

```
#include <stdio.h>
void sort(int a[] , int size){
    int i, j;
    for (i = 0 ;  i <= size-1 ; i++){
        for (j = 0 ;  j < i ; j++){
                            if (a[i] < a[j]){                /* decreasing order */
                                 a[j]   = a[j] + a[j+1] ;
                                 a[j+1] = a[j] - a[j+1] ;
                                 a[j]   = a[j] - a[j+1] ;
                            }
       }
   }
}
int main(){
  int size = 0, i, ropes[1000] ;
  scanf("%d", &size ) ;
  for( i = 0 ; i < size; i++) scanf("%d", &ropes[i]);
  sort(ropes, size);
  int max = ropes[0] ;
  /*******************************************************/
  for( i = 1 ; i < size ; i++ ){
    if ( ropes[i] > ( max / (i+1) ) ) max = ropes[i]*(i+1) ;
    else break ;
  }
  /*******************************************************/
  printf("%d\n", max );
  return 0 ;
}
```

الگوریتم مرتب سازی که در متد sort بکار رفته است مرتب‌سازی حبابی Bubble-Sort نام دارد. شما می

توانستید از هر الگوریتمی که راحت‌تر بودید استفاده کنید .(برای مثال الگوریتم مرتب سازی انتخابی (که در همین تمرین آمده بود) یا مرتب سازی درجی یا مرتب سازی سریع) بهتر است آنرا یاد بگیرید. مضافاً دقت کنید که swap بکاررفته در این الگوریتم از متغیر اضافه استفاده نمی‌کند.

به این شیوه حل مسائل که ما در طی مسئله جواب را به صورت پویا می‌سازیم برنامه نویسی دینامیک Dynamic Programming می گویند. عموم مسئله هایی که باروش دینامیک حل می‌شوند (از پایین به بالا) از سرعت بالاتری نسبت به روش های مشابه بازگشتی (از بالا به پایین) بر خوردارند.

برای تمرین سعی کنید اعداد فیبوناچی را به صورت دینامیک و در درون یک آرایه پیاده‌سازی کنید.
برای تمرین بازهم بیشتر تلاش کنید مسئله کوله‌پشتی صفر-یک را به صورت دینامیک حل کنید.
برای تمرین بیشتر بیشتر سوال زیر را هم با روش بازگشتی و هم با روش دینامیک حل کنید. بحث کنید که چرا روش دینامیک سریعتر به جواب میرسد.
/ چند جمله ای های چبیشف به صورت زیر تعریف می شوند:

$$T_i(x) = \begin{cases} ۱, & i = ۰؛ \\ x, & i = ۱؛ \\ ۲ \times x \times T_{i-۱}(x) - T_{i-۲}(x) & i > ۱. \end{cases}$$

برنامه ای بنویسید که با گرفتن i و x مقدار چند جمله ای چبیشف $T_i(x)$ را حساب کند.

۲ ..... تابع گاما

این سوال تنها یک محاسبه ساده ریاضی لازم داشت :

```
#include <stdio.h>
#include <math.h>

int main(){
    double n ;
    scanf( "%f" , &n );
    float y = sqrt( 2 * 3.1415926 * n) * pow(n,n) * exp( -n ) ;
    printf( "%f" , y );
    return 0 ;
}
```

در صورتی که مایل بودید نمره اضافی بگیرید می بایست چند خطی به کدتان اضافه می کردید:

```
int main()
{
  float n; scanf( "%f" , &n );
  int i ;
  float coef[4] = { 1.0,
                    0.083333 ,
                    0.003472 ,
                   -0.002681
                   };

  float c = 0 ;
  for ( i = 0 ; i < 4 ; i++ ) c += coef[i]*pow(n,-i) ;
  float y = sqrt( 2 * 3.1415926 * n) * pow(n,n) * exp( -n ) * c ;
  printf( "%f" , y );
  return 0 ;
}
```

۳ ..... Selection-Sort

دقیقاً همان شکلی که در صورت سوال گفته شده است:

```
#include <stdio.h>
#include <stdlib.h>
#define n 10

void selectionSort( int array[] , int size ){
  if ( size == 1 ) return ;
  else{
      int maximum = 0 ;
      int i ;
      for ( i = 0 ; i < size ; i++ ){
         if ( array[i] > array[maximum] ){
              maximum = i ;
         }
      }
      int temp = array[maximum] ;
      array[maximum] = array [size-1] ;
      array[size-1] = temp ;
  }
  selectionSort(array , size -1 );
}

int main()
{
  int array[n] , i;
  for ( i = 0 ; i < n ; i++ ) array[i] = rand() ;
  selectionSort(array , n);
  for ( i = 0 ; i < n ; i++ ) printf( "%d \n", array[i] ) ;
  return 0;
}
```

به جابه‌جا کردن دو متغیر توجه کنید. به این عمل swap می گویند. در این مثال از یک متغیر اضافی (temp) استفاده شده بود. می توان همین کار را بدون متغیر اضافی نیز انجام داد: (یاد بگیرید!)

```
int x = ... ;
int y = ... ;
x = x + y ;
y = x - y ;
x = x - y ;
```

به علاوه توجه کنید که در مثال فوق الگوریتم selection-sort به صورت بازگشتی نوشته شده است (به فراخوان selectionSort(array, size-1) دقت نمایید.) همین الگوریتم را می توان با یک حلقه دیگر به شکل غیر بازگشتی پیاده‌سازی کرد. کد زیر این عمل را نشان می دهد:

```
void selectionSort(int array[],int size)
{
  int i, j, min, x;
  for( i=0 ; i < size-1 ; i++ ){
    x = i ;
    min=array[i];
    for( j = i+1 ; j < size ; j++ ){
      if ( min > array[j] ){
         x=j;
         min=array[j];
      }
    }
    int temp = array[i] ;
    array[i] = array[x];
    array[x]=temp;
  }
}
```

در گونه سریعتر این الگوریتم که به shaker sort معروف است در هر مرحله علاوه بر پیدا کردن عضو مینیمم عضو ماکزیمم را نیز میابد. آیا می توانید آنرا هم به شیوه مناسبی پیاده سازی کنید؟

ب)

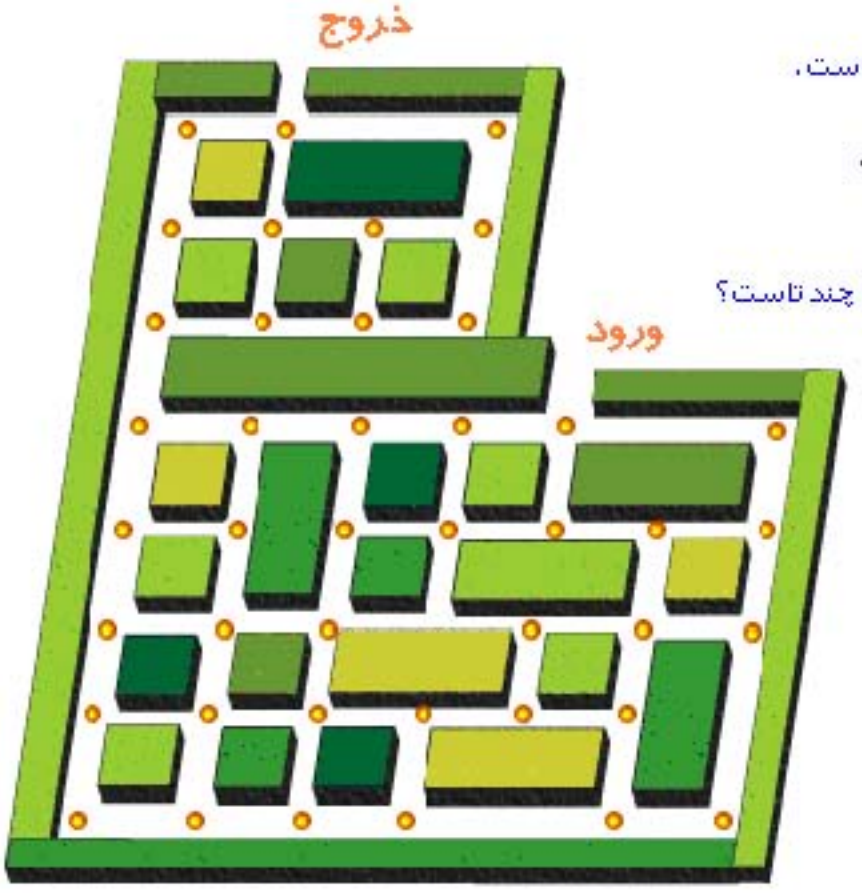
```
#include <stdio.h>
int main()
{
  int x[10],x2[10], i;
  for( i = 0 ; i < 10 ; i++ ) scanf("%d ", &x[i]);
  n %= 10 ;
  for( i = 0 ; i < n ; i++  ) x2[10-n+i] = x[i] ;
  for( i = n ; i < 10 ; i++ ) x2[i-n] = x[i] ;

  for( i = 0 ; i < 10 ; i++ ) printf("%d ", x2[i]);
  return 0 ;
}
```

در این تمرین از یک آرایه اضافی استفاده کردیم. تلاش کنید همین تمرین را بدون استفاده از این آرایه و تنها با یک متغیر اضافی (مثلاً temp) انجام دهید. البته در آنصورت مجبور هستید از حلقه های for تودرتو یا حلقه for همراه با شرط if استفاده کنید.

موفق باشید.

ماز جالب



در ماز روبرو در هر تقاطع يك مانع قرار داده شده است.
ما نمي توانيم از مانع رد شويم اما مي توانيم
مسيرمان را به چپ يا راست تقاطع ادامه دهيم .

مسئله اين است :
كمترين تعداد حركت هاي لازم براي گذر از اين ماز چند تاست؟

بسمه‌تعالی

آشنایی با برنامه سازی ۱۵۳ — ۴۰

تمرین
سری پنجم

زمان تحویل:
«۵ آذر ۸۵»

دقت کنید:[۱]

- **هر تمرین را در یک فایل جداگانه نوشته و تا قبل از نیمه‌شب زمان تحویل به ای-میل درس ارسال کنید.**
- **فیلد** subject **نامه خود را حتماً به صورت زیر ارسال نمایید:** hw5-StudentID **که در آن** StudentID **شماره دانشجویی شماست، برای مثال:** hw5-85123456
- **قبل از فرستادن برنامه از کمپایل شدن و صحت خروجی برنامه هایتان اطمینان پیدا کنید.**
- **ورودی برنامه های شما از صفحه‌کلید و خروجی بر روی صفحه مانیتور خواهد بود.**
- **حل کردن سوال ریسمان‌های پوسیده اختیاری و مشمول نمره اضافه است.**
- **در صورتی که تمرینات شما کپی از روی تمرینات فرد دیگری باشد هر دو (یا چند) نفر نمره منفی دریافت خواهند کرد. به علاوه این وضعیت در نمره تمرینات بعدی و آزمون میان-ترم شما نیز تأثیر منفی خواهد گذاشت.**

ای-میل درس	ce153c@gmail.com
Subject	hw5-????????
Body	خالی بماند.
AttachedFile	RottenRopes.c Gama.c Shakespeare.c array-a.c array-b.c

فهرست:

- عموشکسپیر
- ریسمان های پوسیده
- تابع گاما
- Selection-Sort

موفق باشید.

[۱] محتویات این پرونده با نرم افزار فارسی‌تک‌حروفچینی شده است. نرم افزارفارسی‌تک‌برنامه ای قدتمند و مبتنی بر TEX برای حروفچینی متون ریاضی و غیره می باشد. برای تهیه آن می توانید به آدرس www.farsitex.org مراجعه نمایید.

۰ ..... عموشکسپیر

To Be or Not to Be, Calculate the Result!!!

در نوشتن برنامه های ریاضی فرمت ورودی،خروجی و میزان دقت اعداد اهمیت دارد. برای مثال دستور printf پارامتر های متعددی دارد که به برنامه نویس اجازه سفارشی سازی خروجی را می دهد. تعدادی از آنها در جدول ۱ آورده شده است: (هر یک از این کدها را می توان بعد از علامت ٪ به کاربرد.)

کد	کاربرد
f	برای اعداد اعشاری float
c	حروف
s	رشته‌ها
d	اعداد طبیعی علامت دار
e	نمایش علمی عدد
x یا X	نمایش در مبنای ۱۶
o	نمایش در مبنای ۸

الف) توضیح دهید، تصحیح کنید که چرا کد زیر به درستی کار نمی‌کند؟ (انتظار داریم خروجی عدد ۰.۳۳۳۳۳۳ باشد.) (راه‌نمایی: این کد بیشتر از ۱ خطا دارد.)

```
#include <stdio.h>

int main(){
     float result;          /* Result of the division */
     result = 1/3;          /* Assign result something */
     printf("Result is \%o \n" , result ) ;
     return (0);
}
```

ب) یکی از جملات مشهور انگلیسی عبارتی است که در بالای این تمرین نوشته شده است. این عبارت را به ریاضی ترجمه و با زبان C حاصل این عبارت را در مبنای ۱۶ محاسبه کنید.
یادداشت: در زبان C اعداد مبنای ۱۶ با 0x شروع می شوند. برای مثال:

```
0x2B = 2*16 + 11 = 43
```

ج) یک سوال کلاسیک: برنامه ای بنویسید که بزرگ‌ترین مقسوم‌علیه مشترک ب.م.م. یا gcd دو عدد را حساب نماید.

۱ ..... ریسمان های پوسیده

فرض کنید n طناب با طولهای مساوی داریم و می خواهیم از آنها برای بلند کردن جسم سنگینی استفاده نماییم. بیشترین نیرویی که هر طناب می تواند تحمل نماید مشخص است و اگر نیرویی بیشتر از آن به طناب اعمال شود طناب پاره خواهد شد. برای بلند کردن جسمی به وزن w می توانیم k طناب را به طور موازی به جسم ببندیم به طوری که به هر یک نیروی $\frac{w}{k}$ وارد شود. برای مثال، اگر سه طناب با تحمل ۳، ۱۰ و ۱۵ برای بلند کردن جسمی استفاده کنیم وزن جسم نباید از ۹ نیوتن بیشتر باشد وگرنه ضعیف ترین طناب پاره خواهد شد. اما طناب دوم به تنهایی می تواند وزنه ای ۱۰ نیوتنی را بلند نماید. اما مجموعه طناب ۱۰ و ۱۵ می توانند با هم یک وزنه ۲۰ نیوتنی را تحمل نمایند.

برنامه ای بنویسید که ابتدا n تعداد طناب ها را از ورودی بگیرد و سپس n عدد صحیح را که نیروی تحمل طناب ها است از ورودی دریافت کند. خروجی بیشترین وزنی است که می توان توسط زیرمجموعه دلخواهی‌از این طناب ها بلند کرد به طوری که هیچ طنابی پاره نشود.

مثال:

ورودی:

۲

۱۰ ۱۵

خروجی:

۲۰

۲ ..... تابع گاما

تابع گاما یکی از توابعی است که کاربرد های بسیار زیادی را در ریاضیات مهندسی دارد. تعریف این تابع به صورت زیر است:

$$\Gamma(x) = \int_{۰}^{\infty} t^{x-۱} e^{-t} dt$$

یکی از خواص جالب این تابع این است که $n! = \Gamma(n+۱)$ یعنی تابع گاما در رابطه بازگشتی $\Gamma(x+۱) = x\Gamma(x)$ صدق می کند. این رابطه این توانایی را به ما می دهد تا تابع فاکتوریل (را که بالفطره برای اعداد طبیعی تعریف شده است.) را به اعداد حقیقی تعمیم دهیم. برای مثال داریم: $\Gamma(\frac{۱}{۲}) = \sqrt{\pi}$

بدیهی است که محاسبه عددی این تابع از روی تعریف کار نسبتاً دشواری است. خوشبختانه رابطه ای موسوم به رابطه استرلینگ وجود دارد که این مقدار را تقریب می زند:

$$\Gamma(n+۱) \simeq \sqrt{۲\pi n} n^n e^{-n}$$

ورودی برنامه شما یک عدد حقیقی مثبت است. فرض نمایید این عدد به اندازه کافی کوچک می باشد تا سرریز رخ ندهد. در خروجی گامای این عدد را به کمک تقریب فاکتوریل استرلینگ تقریب بزنید.

مثال:

ورودی:

۶.۱۰

خروجی:

۱۴۲.۴۵۱۹۲۰

یادداشت:

- رابطه دقیق سری مجانبی استرلینگ به صورت زیر است:

$$\Gamma(n+۱) = \sqrt{۲\pi n} n^n e^{-n} \{ ۱ + \frac{۱}{۱۲n} + \frac{۱}{۲۸۸n^۲} - \frac{۱۳۹}{۵۱۸۴۰n^۳} + \ldots \}$$

این رابطه بسیار دقیقی است به طوری که مقدار خطای آن تنها به ازای چند مرحله ابتدایی به کمتر $|\epsilon| < ۲ \times ۱۰^{-۱۰}$ می رسد. پیاده سازی تابع گاما از روی این سری مجانبی مشمول نمره اضافه خواهد بود.

- درباره تابع گاما می توانید در اکثر کتب ریاضی مهندسی مطالب جامعی را پیدا نمایید. برای مثال کتاب «معادلات دیفرانسیل معمولی» از دکتر شادمان و دکتر مهری یا Advance Mathematics for Engineers and Scientists از Murray R. Spiegel مراجع خوبی به حساب می آیند. پیاده سازی دقیقی از این تابع به کمک لگاریتم طبیعی را می توانید در کتاب Numerical Recipes in C از Press et al. ببینید. به کمک لگاریتم طبیعی محدودیت محاسبه اعداد بزرگ (به دلیل سرریزی) تا حد زیادی از بین می رود.

۳ ..... Selection-Sort

الف) مرتب سازی پای ثابت همه تمرین های برنامه‌نویسی است.در این سوال می بایست الگوریتم Selection-Sort را برای یک آرایه int پیاده‌سازی کنید. این الگوریتم به شرح زیر است:

۱) کوچکترین عنصر را در آرایه بیابید و آنرا با مکان نخست جابه‌جا کنید.

۲) دومین عنصر کوچک را در آرایه بیابید و آن را با مکان دوم جابه‌جا کنید.

۳) این شیوه را تا آخر ادامه دهید.

برنامه ای بنویسید که یک آرایه ۱۰ عضوی با درایه های تصادفی ایجاد نموده و آن را مرتب کند.

ب) برنامه ای بنویسید که یک آرایه ۱۰ عضوی از اعداد طبیعی ایجاد و اعضای آن را از ورودی بخواند. سپس با خواندن عدد طبیعی n از ورودی اعضای این آرایه را n واحد به سمت چپ دوران (شیفت) دهد. توجه کنید ممکن است n از ۱۰ بزرگتر باشد. اگر به ابتدای آرایه رسیدید از انتها شروع کنید. آرایه تبدیل شده را به خروجی ارسال کنید.

مثال:
ورودی
۰ ۱ ۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹
۲

خروجی
۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹ ۰ ۱

یادداشت: برای ایجاد اعداد تصادفی می توانید از تابع ()rand در هدرفایل stdlib.h استفاده نمایید.

Language shapes the way we think,
and determines what we can think about.
B. L. Whorf

خسته نباشید.