ایلیا و مشکلات با اعداد کسری

ایلیا قصد داشته که توی برنامه هایی که مینویسه، از اعداد کسری استفاده کنه، اما از اونجایی که کار با float و مخصوصا double به شدت گاهی اذیت میکنه، تصمیم گرفته از شما کمک بگیره تا بتونه راحت و بی دردسر کسر هارو نگه داره! در این سوال قصد داریم مکانیزم تعریف اعداد گویا را با کمک **structure** ها انجام دهیم.

میدونید که هر عدد گویا از یک صورت و یک مخرج تشکیل شده است. پس structای تعریف کنید که دو فیلد صحیح صورت و مخرج داشته باشد به عنوان مثال:

```c
struct Rational {
    int a;
    int b;
};
```

که تعریف کسر به شکل $\frac{a}{b}$ است.

نکته: ساختار داده شده، الزاما ساختار مورد استفاده ی شما نخواهد بود و ممکن است نیاز به تغییرات داشته باشد.

حال توابع زیر را برای محاسبه ی اعمال کسری باید پیاده سازی کنید:

Copy C

```c
void getRational(Rational *i);
```

این تابع صورت و مخرج کسر را از ورودی دریافت کرده و در آرگومان ورودی ذخیره میکند.

```c
void print(Rational i);
```

این تابع، آرگومان ورودی را به شکل a/b در خروجی نمایش می دهد.

```c
void simplify(Rational *i);
```

این تابع کسر ورودی را به ساده ترین حالت ممکن تبدیل می کند.

```
Rational add(Rational q1 , Rational q2);
```

این تابع دو کسر ورودی را با هم جمع کرده و به عنوان خروجی تابع بر میگرداند.

```
Rational subtract(Rational q1 , Rational q2);
```

این تابع کسر دوم را از کسر اول کم کرده و آن را به عنوان خروجی تابع بر میگرداند.

```
Rational multiply(Rational q1 , Rational q2);
```

این تابع دو کسر را در هم ضرب میکند و به عنوان خروجی تابع بر میگرداند.

```
void reverse(Rational *i);
```

این تابع کسر ورودی را معکوس می کند.

```
Rational divide(Rational q1 , Rational q2);
```

این تابع دو کسر را بر هم تقسیم می کند.

ورودی و خروجی

پیاده کردن main برنامه اختیاری و به عهده ی خودتان است و تکالیف به صورت دستی تصحیح خواهد شد، اما دقت کنید در main حداقل یکبار تمام توابع خود را تست کنید. نکته ی مهم در تصحیح این است که توابع پیاده سازی شده، بهینه و جامع باشد؛ طوری که پاسخگوی تمام تست کیس ها باشد.

تذکر:

- صورت و مخرج کسر ها اعداد صحیح خواهد بود.

- به Call by reference یا Call by value بودن توابع دقت کنید.
- تعریف توابع بالا اجباریه، اما میتونین خودتون برای راحتی کار توابع دیگه ای هم تعریف کنید که کارتون راحت باشه، یادتون باشه اینجا نمیخوایم کدمون فقط کار کنه، میخوایم بهینه کد بزنیم، پس حواستون به حجم کدتون باشه.(مثلا تعریف یک تابع ک.م.م حجم کدتون رو خیلی کمتر میکنه)
- سعی در کنترل دستی خطاها با شرط داشته باشید، حالات استثنا و غیر ممکن رو در نظر داشته باشید:)

به نمونه پیاده سازی Main زیر دقت کنید:

```
int main(){
Rational Q1 , Q2 , Q3;
getRational(&Q1);
//فرض کنید کاربر عدد 2 و 5 را وارد میکند
getRational(&Q2);
//فرض کنید کاربر عدد 6 و 3 را وارد میکند

Q3 = add(Q1 , Q2);
print(Q3);
// باید مقدار 36/15 پرینت شود.
simplify(&Q3);
print(Q3);
// باید مقدار 12/5 پرینت شود.
}
```

ضرب ماتریس

در این سوال به عنوان ورودی دو ماتریس داده می شود و به عنوان خروجی باید ضرب دو ماتریس را حساب کنید.

در سوال ابتدا سه عدد داده می شود عدد اول و دوم به ترتیب سطر و ستون ماتریس اول و عدد دوم و سوم نیز سطر و ستون ماتریس سوم می باشد.

دقت شود از آنجایی که اندازه ی ماتریس مشخص نیست بنابراین حتما باید آرایه دو بعدی خود را به صورت پویا تعریف کنید.

در سوال باید توابع زیر را پیاده سازی نمایید.

```
void CreateMatrix( /*input*/ )
```

در این تابع به عنوان ورودی ماتریس و تعداد سطر و ستونش را گرفته و حافظه ی لازم را به آن ماتریس تخصیص می دهید.

```
void FillMatrix( /*input*/ )
```

در این تابع به عنوان ورودی ماتریس و تعداد سطر و ستونش را گرفته و سپس درون تابع از کاربر ورودی گرفته و ماتریس را پر می کنید.

```
void PrintMatrix( /*input*/)
```

در این تابع به عنوان ورودی ماتریس و تعداد سطر و ستونش را گرفته و آن را چاپ نمایید.

این سه تا تابع را حتما باید پیاده سازی نمایید. براساس سوال نیز می توانید توابع دیگری نیز پیاده سازی کنید تا کد شما کوتاه تر شود. (تابعی مانند حساب کردن ضرب دو تا ماتریس و ...)

نکته مهم:

از آنجایی که داریم آرایه دو بعدی خود را به صورت پویا تعریف می کنیم پس حتما حواستان به memory leak باشد. همچنین این تمرین دستی هم تصحیح می شود و 100 گرفتن در تست کیس ها نشانه ی 100 گرفتن در این سوال نمی باشد و توابعی که می نویسید خوانده می شود.

راهنمایی: برای ورودی توابع به اینکه به صورت call by value ورودی بگیرین یا call by reference توجه نمایید.

ورودی

در خط اول سه عدد داده می شود که نشان دهنده ی سطر و ستون دو ماترس می باشد. در خطوط بعدی نیز مقادیر درون ماتریس ها داده می شود.

خروجی

حاصل ضرب دو ماتریس را به فرمتی که در پایین نشان داده شده چاپ نمایید.

مثال

ورودی نمونه ۱

```
2 3 4
2 1 4
5 4 1
1 2 3 5
5 4 3 1
6 3 2 1
```

خروجی نمونه ۱

```
31 20 17 15
```

```
31 29 29 30
```

با توجه به خط اول ماتریس اول یک ماتریس 2 در 3 می باشد و ماتریس دوم نیز یک ماتریس 3 در 4 می باشد. سپس در خطوط بعدی مقادیر دو ماتریس داده شده است. در خروجی نیز با توجه به ماتریس های اولیه مون خروجی در یک ماتریس 2 در 4 حساب شده است.

ورودی نمونه ۲

```
3 3 3
1 2 3
4 5 6
7 8 9
9 8 7
6 5 4
3 2 1
```

خروجی نمونه ۲

```
30 24 18
84 69 54
138 114 90
```

وکتور خودمه :)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

در این سؤال قصد داریم یک لیست پیوندی یک طرفه به همراه توابع مهم آن را پیاده‌سازی کنیم.

هر Node از این لیست پیوندی در واقع یک struct به صورت زیر میباشد :

```
typdef struct Node{
    int data;
    Node *next;
}Node;
```

همچنین شما مجاز به استفاده از یک هد گلوبال در برنامه هستید.

```
Node head = {0, nullptr};
```

این لینک لیست باید دارای قابلیت های زیر به صورت کامل باشد

● Node* creartNode(int data)

این تابع برای ایجاد یک Node با توجه به data ورودی استفاده شده و اشاره گر مربوطه را به عنوان خروجی بر می گرداند.

● void printNode(Node* node)

از این تابع برای چاپ data یک Node ورودی استفاده می شود.

برای لیست پیوندی خود توابع زیر را پیاده‌سازی نمایید:

● void push_front(int data)

این تابع ابتدا یک Node جدید با توجه به ورودی data ساخته و سپس آن را به ابتدای لیست پیوندی اضافه می نماید.

- void push_back(int data)

این تابع ابتدا یک Node جدید با توجه به ورودی data ایجاد کرده و سپس آن را به انتهای لیست پیوندی اضافه می نماید.

- void pop_front()

این تابع اولین Node (در صورت وجود) از لیست پیوندی را حذف می نماید.

- void pop_back()

این تابع آخرین Node (در صورت وجود) از لیست پیوندی را حذف می نماید.

- void insert (int data, int index)

این تابع برای اضافه کردن یک Node به لیست پیوندی در موقعیت دلخواه استفاده می شود. به این صورت که درون این تابع ابتدا یک Node جدید با توجه به data ایجاد و در موقعیت index به لیست پیوندی اضافه می شود. (در صورتی که index بیشتر یا مساوی سایز وکتور باشد، باید پیغام زیر چاپ شده و از تابع خارج شویم.)

> invalid input size!

- void delet(int index)

از این تابع برای حذف کردن یک Node از موقعیت دلخواهی از لیست پیوندی استفاده می شود. (در صورتی که index بیشتر یا مساوی سایز وکتور باشد، باید پیغام زیر چاپ شده و از تابع خارج شویم.)

> invalid input size!

- int search(int data)

این تابع برای جست و جوی یک مقدار دلخواه در لیست پیوندی استفاده شده و اندیس مربوطه به عنوان خروجی برگردانده می شود. اگر مقدار data در آرایه یافت نشد، ۱- برگردانده شود.

- Node* max()

این تابع اشاره گری به Node با بیشترین مقدار data را بر می گرداند.

- int average()

این تابع میانگین مقادیر (data) نوهای لیست پیوندی را بر میگرداند.

- void swap(int index1, int index2)

از این تابع برای جا به جایی data مربوط به دو Node با اندیس های index1 و index2 استفاده می شود.

- void print()

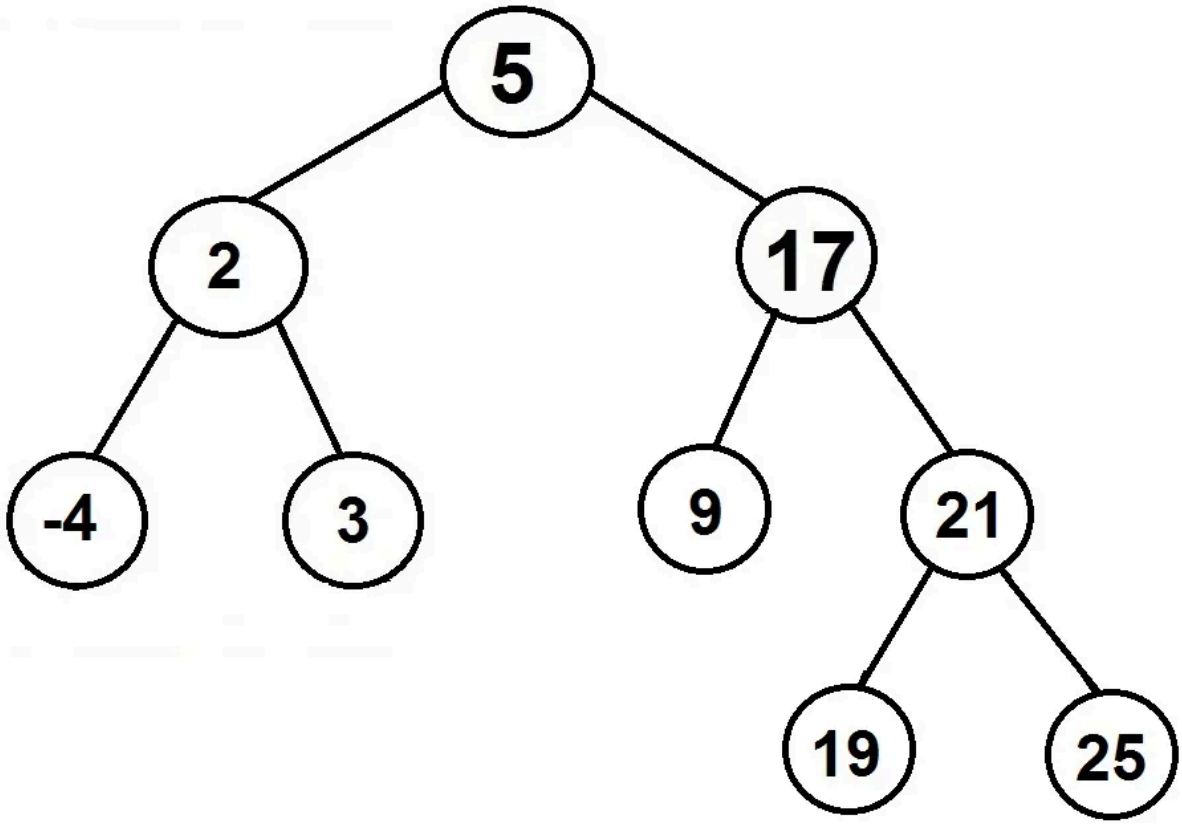
این تابع تمام مقادیر (data) مربوط به Node ها را از ابتدا تا انتها چاپ می نماید.

** برنامه‌ی شما نیاز به تابع main ندارد اما اکیدا توصیه می‌شود که یک بار برنامتون رو تست کنین در حالت های مختلف چون قراره به‌صورت دستی کدتون بازنگری بشه.

درخت باینری (امتیازی)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

درخت باينري درختي که هر گره حداکثر دو فرزند داشته باشد. به عنوان مثال شکل راست درخت باينري است ولي شکل سمت چپ درخت باينري نيست.



توجه کنيد که در يك درخت باينري فرزند چپ هر گره کوچکتر از آن و فرزند راست هر گره بزرگتر از آن گره می‌باشد.

حال اگر درخت جستجوي باينري را به صورتي پيمايش کنيم که از ريشه شروع کرده و براي هر گره:

۱. فرزند چپ را چاپ کنيم.

۲. خود گره چاپ کنیم.

۳. فرزند راست را چاپ کنیم. در این صورت اعداد را به صورت صعودي چاپ کرده ایم. (اصطلاحاً به این نوع پیمایش، پیمایش inorder گفته میشود.) در این سؤال باید یك درخت جستجوي باینري را پیاده سازي کنید. ابتدا عدد n از کاربر گرفته میشود که نشان دهنده تعداد عناصر است. سپس به تعداد n ،عدد از کاربر بگیرید. از روي این اعداد درخت جستجوي باینري بسازید، سپس آن را به صورت inorder پیمایش کنید تا عناصر به صورت صعودي چاپ شوند.

مثال

ورودی نمونه

```
5
3 2 4 5 1
```

خروجی نمونه

```
1 2 3 4 5
```