لیست از مدل لینک شده

لینک لیستی از node که از استراکت زیر ساخته شده است بنویسید:

Copy

```
struct node{
int data;
node* next;
};
```

سپس برای این لینک لیست تابع

```
void add(int input);
```

برای اضافه کردن نود جدید به آخر لیست

تابع delete برای پاک کردن آخرین نود لینک لیست

```
void delete (int deleteNode);
```

تابع سرچ به صورت

```
node* search(int check);
```

که آدرس نود را بر می گرداند و در صورتی که پیدا نکند آدرس آخرین نود را بر می گرداند

تابع insertIn که با گرفتن اندیس یک نود جدید در محل جدید اضافه می کند

```
void insertIn(int where, int input);
```

و تابع print که کل لینک لیست شما را چاپ کند

```
void print ();
```

این تمرین رو طوری بزنید که در تمرین های بعدی استفاده کنید. و حتما مدیریت حافظه رو داشته باید و خانه های اضافه را free کنید.

مسافرت طولانی

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

ما برای تعطیلات به کشور آتینگا رفتیم و قصد داریم از شهرهای مختلف این کشور بازدید کنیم.

ویژگی خاص مسیرها در این کشور این است که از هر شهر آن به هر شهر دیگر میتوان سفر کرد ولی از هر شهر به هر شهر دیگر دقیقا یک مسیر خاص وجود دارد.

باتوجه به ورودی‌های مسئله که به صورت زیر داده می‌شوند، طول بلندترین مسیری که می‌توان در این کشور سفر کرد بدون این که از هیچ شهری بیش از یک بار عبور کرد را چاپ کنید(طول مسیر را تعداد شهرهایی که در این سفر خواهید دید فرض کنید).

ورودی

ابتدا عدد n داده می‌شود که تعداد شهرهای این کشور است. سپس به طور متوالی در $n-1$ سطر در هر سطر دو عدد با یک فاصله بین‌شان داده می‌شود که این اعداد بین 1 تا n اند و هر عدد مشخص‌کننده‌ی یک شهر می‌باشد، و هر سطر ورودی مشخص می‌کند بین دو شهر نام برده شده جاده‌ای احداث شده است.

$$1 \leq n \leq 10^6$$

خروجی

تنها یک عدد است که طول بلندترین مسیر در این کشور می‌باشد(طول مسیر را تعداد شهرها مشخص می‌کند.)

مثال

نمونه ورودی

```
5
1 2
2 3
```

```
3 4
4 5
```

نمونه خروجی

```
5
```

وکتور خودمه :)

**در این سوال قصد داریم یک لیست پیوندی یک طرفه به همراه توابع مهم آن را پیاده سازی کنیم.

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

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

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 پیاده سازی 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 insert)

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.

- void delete(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 آرایه باشد، باید پیغامی مناسب چاپ شده و از تابع خارج شویم.)

- 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 sort()

از این تابع برای مرتب سازی صعودی Node های لیست پیوندی با توجه به data آن ها استفاده می شود.

راهنمایی: می توانید در صورت نیاز از تابع swap پیاده شده در قسمت قبل استفاده نمایید.

- void print()

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

- void reversePrint()

عملکرد این تابع مشابه تابع بالاست فقط ترتیب data های چاپ شده در این تابع از آخر به اول است.

باینری سرچ تیری

درخت باينري درختي كه كه هر گره حداكثر دو فرزند داشته باشد. به عنوان مثال شكل سمت چپ درخت باينري است ولي شكل سمت راست درخت باينري نيست. درخت جستجوي باينري يك درخت باينري است به طوري كه فرزند چپ هر گره كوچكتر از آن و فرزند راست هر گره بزرگتر از آن باشد. حال اگر درخت جستجوي باينري را به صورتي پيمايش كنيم كه از ريشه شروع كرده و براي هر گره:

1 .فرزند چپ را چاپ كنيم.

2 .خود گره چاپ كنيم.

3 .فرزند راست را چاپ كنيم.

در اين صورت اعداد را به صورت صعودي چاپ كرده ايم. (اصطلاحاً به اين نوع پيمايش، پيمايش inorder گفته ميشود.) در اين سوال بايد يك درخت جستجوي باينري را پياده سازي كنيد. ابتدا عدد n از كاربر گرفته ميشود كه نشان دهنده تعداد عناصر است. سپس به تعداد n ،عدد از كاربر بگيريد. از روي اين اعداد درخت جستجوي باينري بسازيد، سپس آن را به صورت inorder پيمايش كنيد تا عناصر به صورت صعودي چاپ شوند. مثال:

مثال

ورودی نمونه ۱

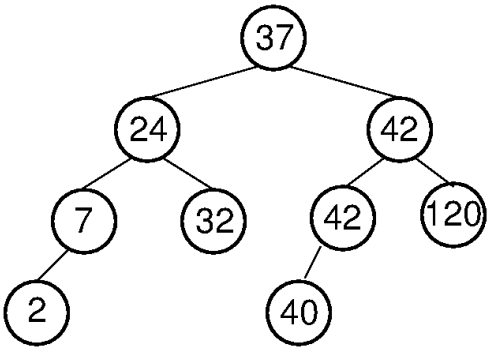
```
5
3 2 4 5 1
```

خروجی نمونه ۱

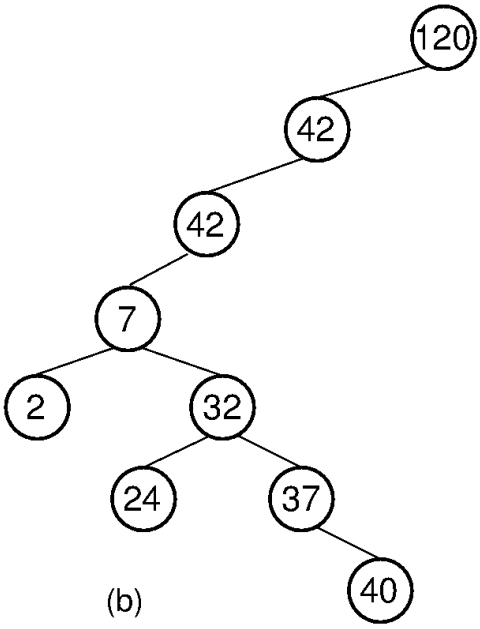
```
1 2 3 4 5
```

به فرمت ورودي و خروجي دقت كنيد. (بعد از چاپ هر عدد يك space نيز چاپ كنيد.)

چند نمونه درخت باینری جستجو:



(a)



(b)