مقدمات درخت

1. تعریف درخت را بنویسید و تفاوت آن را با گراف بیان کنید. همچنین توضیح دهید که چرا درخت‌ها معمولاً به عنوان ساختار داده مناسب برای نمایش سلسله‌مراتب (hierarchy) استفاده می‌شوند.

2. در یک درخت دودویی کامل (Complete Binary Tree) با ۱۰۰ گره:

۱. بیشینه تعداد گره‌های برگ چقدر است؟

۲. حداقل و حداکثر ارتفاع درخت را مشخص کنید.

۳. تعداد گره‌های داخلی (non-leaf nodes) را تقریب بزنید و توضیح دهید چرا این تعداد به‌صورت تقریبی برابر با نصف کل گره‌هاست.

3.درخت دودویی‌ای را فرض کنید که پیمایش‌های زیر را دارد:

- **پیمایش درمیان‌وندی (Inorder):** `D B E A F C`
- **پیمایش پس‌وندی (Postorder):** `D E B F C A`

سوال:

- الف) ریشه‌ی درخت را پیدا کنید.
- ب) پیمایش پیش‌وندی (Preorder) این درخت را بنویسید.
- پ) شکل تقریبی درخت دودویی را رسم کنید.

درخت عمیق

درخت دودویی به صورت زیر تعریف شده است:

```
struct TreeNode {
    int value;
    TreeNode* left;
    TreeNode* right;
};
```

یک تابع بازگشتی بنویسید که ارتفاع درخت دودویی را محاسبه کند. ارتفاع یک درخت، تعداد یال‌های مسیر بلندترین بین ریشه تا یک برگ است.

ورودی

اشاره‌گر به ریشه درخت دودویی

خروجی:

عدد صحیح برابر با ارتفاع درخت (اگر درخت فقط شامل یک گره باشد، ارتفاع آن صفر است.)

نکته:

برای حل این مسئله از پیمایش پس‌وندی (Postorder Traversal) استفاده کنید، چون ابتدا باید ارتفاع فرزندان چپ و راست محاسبه شود.

درخت عمومی

در یک درخت عمومی (هر گره می‌تواند تعداد دلخواهی فرزند داشته باشد)، رابطه‌ی بین تعداد گره‌ها (n)، تعداد برگ‌ها (L)، و درجه‌ی گره‌ها چیست؟

۱. تعریف‌های زیر را به‌طور کامل بنویسید:

- گره داخلی (Internal Node)
- گره برگ (Leaf Node)
- درجه یک گره (Degree of a Node)
- درجه‌ی درخت (Degree of Tree)

۲. اگر مجموع درجات همه گره‌های یک درخت برابر با S باشد، رابطه‌ای بین S و تعداد کل یال‌ها (edges) در درخت بیان کنید و دلیل آن را توضیح دهید.

۳. در یک درخت با n گره، چند یال وجود دارد؟ اثبات کنید.

۴. نشان دهید که اگر درختی دارای فقط یک ریشه و چندین گره داخلی و برگ باشد، فرمول زیر برقرار است:

$$n = I + L$$

که در آن:

- n: تعداد کل گره‌ها
- I: تعداد گره‌های داخلی
- L: تعداد گره‌های برگ

۵. اگر هر گره داخلی دقیقاً k فرزند داشته باشد (درخت $k-ary$ کامل)، فرمولی برای تعداد برگ‌ها (L) به‌دست آورید.

درخت زنجیری

درخت دودویی کاملی را در نظر بگیرید که به‌صورت دینامیک و با استفاده از ساختار گره‌های اشاره‌گری (linked list) پیاده‌سازی شده است. تنها اشاره‌گر به ریشه درخت در اختیار شماست.

```
struct Node {
    char data;
    Node* left;
    Node* right;

    Node(char val){
     data(val);
     left(nullptr);
     right(nullptr)
     }
};
```

الف)

الگوریتمی برای یافتن **اولین جای خالی** (مکان مناسب برای درج گره جدید) در این درخت بنویسید. روش کار الگوریتم را توضیح دهید و دلیل استفاده از پیمایش سطحی (Breadth-First / Level Order) را بیان کنید.

```
Node* FindFirstEmpty(Node* root){
// تابعی برای پیدا کردن اولین جای خالی جهت درج گره جدید در درخت دودویی کامل
}
```

ب)

الگوریتمی برای یافتن **آخرین برگ** در درخت بنویسید. روش پیمایش را مشخص کرده و توضیح دهید چرا آخرین گره مشاهده‌شده در پیمایش سطحی معادل آخرین برگ در درخت دودویی کامل است.

```
Node* FindLastLeaf(Node* root){
// تابعی برای پیدا کردن آخرین برگ در درخت دودویی کامل

```

```
}
```

توضیح:

- هر دو تابع از **پیمایش سطحی (Level-Order Traversal)** استفاده می‌کنند و برای این کار معمولاً از **صف (queue)** کمک می‌گیریم.
- در تابع `findFirstEmpty` به‌دنبال اولین گره‌ای هستیم که **فرزند چپ یا راست ندارد**.
- در تابع `findLastLeaf` تا پایان پیمایش سطحی جلو می‌رویم و آخرین گره‌ای که از صف خارج می‌شود همان **آخرین برگ** است.

اهرام کاغذی

به سوالات هرمی زیر پاسخ دهید.

1.1) هیپ (**Heap**) چیست؟

1.2) به طور کلی چند نوع هیپ داریم؟ هرکدام را توضیح دهید.

2) آرایه {91, 9, 40, 22, 15-, 1} را به کمک مرتب سازی هرمی (**Heap Sort**) گام به گام و با رسم شکل مرتب کنید.

3) کدام یک از آرایه های زیر، یک هیپ را تشکیل می دهند ؟

A) {23, 17, 14, 6, 13, 10, 1, 12, 7, 5}

B) {23, 17, 14, 6, 13, 10, 1, 5, 7, 12}

C) {23, 17, 14, 7, 13, 10, 1, 5, 6, 12}

D) {23, 17, 14, 7, 13, 10, 1, 12, 5, 7}

ساخت اهرام مصر

ساختمان داده Heap را در cpp پیاده سازی کنید که شامل عملیات های (Operations) زیر باشد:

- **Heapify** → Process to rearrange the heap in order to maintain heap-property.
- **Insertion** → Add a new item in the heap.
- **Deletion** → Delete an item from the heap.
- **Extract Min-Max** → Returning and deleting the maximum or minimum element in max-heap and min-heap respectively.
- **INCREASE-KEY** →

هرم بزرگان قبیله

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

یک آرایه از اعداد صحیح به نام arr با اندازه n و یک عدد صحیح مثبت K داده شده است. وظیفه شما پیدا کردن kامین عنصر بزرگ در آرایه می‌باشد (توجه داشته باشید که منظور از kامین عنصر بزرگ، عنصر kام در ترتیب نزولی آرایه است و نه kامین عنصر یکتا).

توجه داشته باشید هدف سوال استفاده از **Heap Sort** و **پیاده سازی کامل** آن است. (استفاده از **priority_queue** غیر مجاز است.)

ورودی

ورودی برنامه‌ی شما باید شامل ۲ خط باشد:

- در خط اول ورودی دو عدد طبیعی n و k با فاصله از هم آمده است.

$$1 \leq n, k \leq 100$$

- در خط بعدی آرایه ای از اعداد با طول n دریافت می شود.

خروجی

خروجی تنها شامل یک خط است که kامین عنصر بزرگ در آرایه است.

ورودی نمونه ۱

```
7 3
1 23 12 9 30 2 50
```

خروجی نمونه ۱

```
23
```

ورودی نمونه ۲

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

خروجی نمونه ۲

```
5
```

ورودی نمونه ۳

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

خروجی نمونه ۳

```
4
```

k تا از کیش خود

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

آرایه‌ای مرتب به نام **arr** و یک مقدار **x** داده شده است. **k** عنصری که به **x** نزدیک‌ترین هستند را در آرایه پیدا کنید. توجه داشته باشید که اگر عنصر **x** در آرایه موجود باشد، نباید در خروجی گنجانده شود و فقط سایر عناصر نزدیک پذیرفته می شوند.

نکته: استفاده از **priority_queue** مجاز است.

ورودی

ورودی برنامه‌ی شما باید شامل 2 خط باشد:

- در خط اول ورودی سه عدد طبیعی $k، n$ و x با فاصله از هم آمده است.

$$1 \leq n, k \leq 100$$

$$k \leq n$$

- در خط بعدی آرایه ای از اعداد با طول n دریافت می شود.

خروجی

خروجی شامل یک خط از **k** عنصر است که به **x** نزدیک ترین هستند.

ورودی نمونه ۱

```
13 4 35
12 16 22 30 35 39 42 45 48 50 53 55 56
```

خروجی نمونه ۱

```
30 39 42 45
```

ورودی نمونه ۲

```
5 2 4
1 3 4 10 12
```

خروجی نمونه ۲

```
3 1
```

ورودی نمونه ۳

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

خروجی نمونه ۳

```
4 2 5 1
```

ورودی نمونه ۴

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

خروجی نمونه ۴

1 1 2 3

آمیختن گروهان

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

با توجه به **k** لیست پیوندی مرتب (**صعودی**) با اندازه‌های متفاوت، وظیفه شما ادغام همه‌ی آن‌ها به گونه‌ای است که ترتیب آن‌ها حفظ شود.

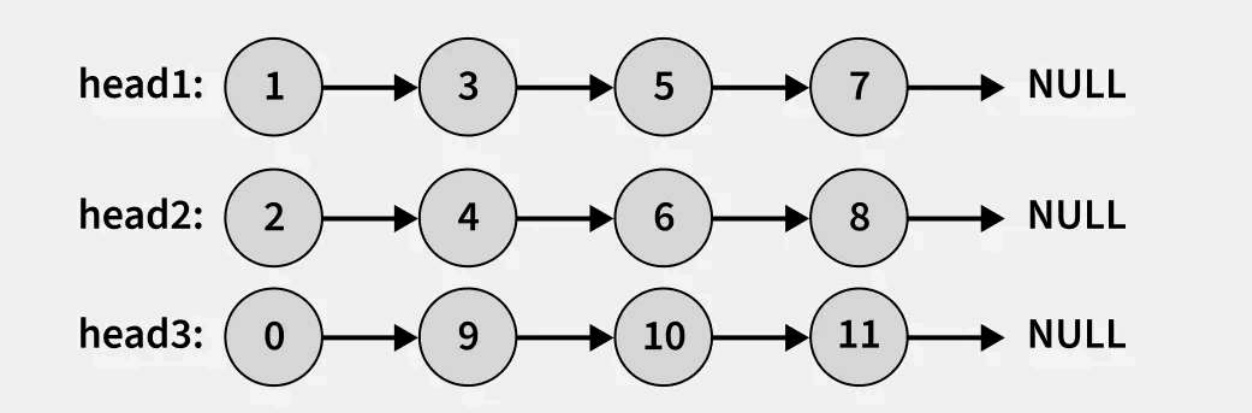
ورودی

ورودی برنامه‌ی شما باید حداقل شامل ۲ خط و حداکثر شامل **k** خط باشد که در هر خط لیست پیوندی مرتب (صعودی) قرار دارد که به صورت **Hard Code** مقدار دهی اولیه می‌شوند.

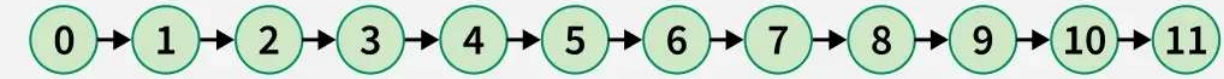
خروجی

خروجی تنها شامل یک خط است که در آن لیست پیوندی ادغام شده (**مرتب به صورت صعودی**) آمده است.

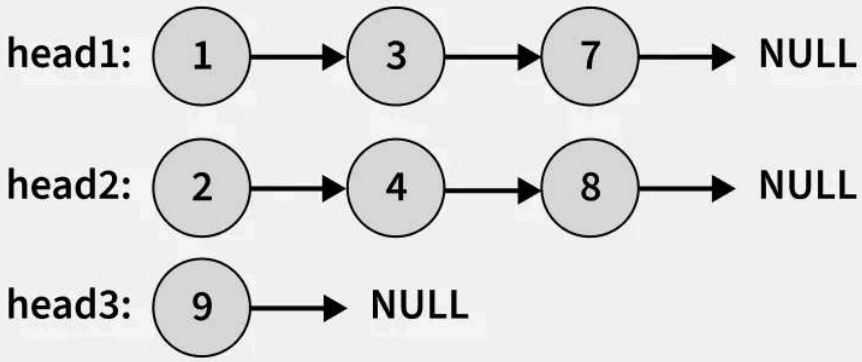
ورودی نمونه ۱



خروجی نمونه ۱



ورودی نمونه ۲



خروجی نمونه ۱