تابع خفن

توی این سوال قراره کار های جالبی انجام بدیم که کمکی به c++ کرده باشیم!

فرض کنید تابعی دارید که میتونید برای هر نوع متغیری اون رو صدا بزنید!

خب بریم سراغ خود تابع:

فرض کنید تابع دو ورودی میگیره که هرکدام از این ورودی ها میتونن حالت های زیر را داشته باشند:

- آرایه فقط از جنس های (int float)
- عدد فقط از جنس های (int float)
- رشته
- بولین (bool)

خروجی تابع به ازای ورودی های مختلف فرق داره و:

۱. اگر یکی از ورودی ها رشته باشد باید دو ورودی به یکدیگر چسبانده شوند و به صورت رشته return شود.
۲. اگر ورودی ها آرایه باشند باید دو آرایه را به یکدیگر چسبانده و آرایه جدید را return کنید. ممکن است یکی از متغیر ها آرایه نباشد (ولی از جنس همان ارایه است) با توجه به اول یا دوم بودن این متغیر باید آن را به اول یا آخر آرایه داده شده اضافه کنید و آرایه جدید را return کنید.
۳. اگر ورودی ها بولین باشند باید دو ورودی را باهم and کرده و return کنید.
۴. اگر ورودی ها عدد باشند باید دو عدد را جمع کنید و return کنید.

نکته

حالت های بالا تنها حالت ها ممکن نیستند و ممکن است حالت های دیگری از ترکیب کردن ورودی های تابع به دست بیاید در صورت پیاده سازی حالت های دیگر به شما نمره اضافه تعلق میگیرد.

سانسورچی

ایلیا قراره به عنوان سانسورچی توی کوئرا فعالیت کنه و باز گیر کرده، مسئولین کوئرا بخاطر محدودیت حافظه روی سرورهاشون، گفتن که نمیتونن برای هر متن یه حافظه ی جدای موقت به ایلیا بدن و ایلیا مجبوره سانسوراش رو روی خود متن در همون لحظه انجام بده!

ورودی

در خط اول تعداد کلمات ممنوعه به شما داده میشه. در خط دوم کلمه(های) ممنوعه که با , جدا شدن بهتون داده میشه. در خط سوم یک متن به شما داده میشه که باید کلمات ممنوعه ازش حذف شه و به جاش، به تعدادِ دو برابرِ حروفش علامت ستاره چاپ کنید. مثلا برای کلمه ی iliya باید 10تا ستاره چاپ شه.

خروجی

در خروجی ، متن سانسور شده چاپ می شود.

ورودی نمونه ۱

```
1
spam
in yek math haavi kalameye spam ast, ma az spam bizaarim;
```

خروجی نمونه ۱

```
in yek math haavi kalameye ******** ast, ma az ******** bizaarim;
```

نکته: مجاز به استفاده از رشته ی کمکی نیستید و باید روی رشته ی اصلی که ورودی روی آن ذخیره می شود، تغییرات رو اعمال کنید.

HANGMAN

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

این لینک ، روند بازی رو کامل توضیح داده :

https://www.youtube.com/watch?v=leW9ZotUVYo

خب همونطور که دیدین ، اول یک نفر یک کلمه ای رو انتخاب میکنه ، بعد ما تعداد حروف این کلمه رو به بقیه نفرات نشون میدیم ، و اونا باید حدس بزنن چه حروفی در این کلمه وجود داره ، اما با هر حدس غلط ، "منِ" بازی ما یه قدم به "هنگ" نزدیک تر میشه :))

اول بازی ، یک کلمه دلخواه رو به صورت main argument از کاربر میگیریم و در هر مرحله باید محتوایی که در فیلم یوتیوب ، روی تخته کشیده میشد رو نشون بدیم:

۱. چقدر از کلمه درست پر شده ،
۲. حدس های اشتباه چیا بودن و
۳. چقدر از آدم ما تا الان کشیده شده. و در پایان بازی هم نشون میدیم که فرد برنده شده یا نه.

- حتما سعی کنین طراحی کدتون به صورت **functional** باشه.
- حواستون باشه که ، حروف بزرگ و کوچک هیچ فرقی باهم ندارن و اونارو **باید یکی فرض کنین**.

بخش امتیازی :

- میتونین یه گزینه داشته باشین که در هر مرحله از بازی ، کاربر بتونه کلمه رو به صورت یکجا وارد کنه و اگه درست حدس زده بود برنده بشه. (طبق قوانین بازی اگه کاربر در این شرایط حدس اشتباهی بزنه، بازی براش تموم میشه و میبازه)

وکتور خودمه++ =)

پیاده‌سازی

سؤال وکتور خودمه رو ک منطقا یادتونه :) اینبار میخوایم ورژن جدیدتری از این سؤال رو داشته باشیم. در این سؤال قصد داریم یک لیست پیوندی یک طرفه به همراه توابع مهم آن را پیاده‌سازی کنیم.

در ابتدا این وکتور (لینکدلیست خودمون) باید دوطرفه بشه. یعنی علاوه بر پوینتر next که به عنصر بعدی اشاره میکنه یه پوینتر preview هم داشته باشه که به عنصر قبلی اشاره میکنه.

در قدم دوم قراره این استراکچر به‌صورت تمپلیت تعریف بشه که هرنوع داده‌ای بشه توش ذخیره کرد. ((تضمین می‌شود تمام نودها در هر تست کیس از یک نوع خواهند بود))

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

```
template <typename T>
struct Node
{
    T data;
    Node *next;
    Node *preview;
};
```

همچنین شما مجاز به استفاده از یک هد گلوبال در برنامه **نیستید** ((چون نمیدونین تایپش چیه خب😅)).

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

-

```
template <typename T>
Node* creartNode(T data)
```

این تابع برای ایجاد یک Node با توجه به data ورودی استفاده شده و اشاره گر مربوطه را به عنوان خروجی برمی گرداند ((توجه شود این تابع باید تمپلیت باشد)).

●

```
template <typename T>
void printNode(Node<T>* node)
```

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

توجه شود این دوتابع مربوط به استراکت Node شما هستند لذا اگر توی بلاک خود استراکت بذارین هم ثواب دنیا رو داره و هم آخرت =))

همچنین برای لیست پیوندی خود توابع زیر را پیاده‌سازی نمایید:

● void push_front(T 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(T 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 (T data, unsigned 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(unsinged 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(T 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 ((توجه شود که اگر نوع دیتای نود از نوع عددی نبود، اگر کاراکتری بود باید براساس حروف الفبا کداسکی آنها را به عنوان مقدار مقایسه درنظر بگیرد و اگر داده از نوع رشته بود باید طول رشته را به عنوان مقدار مقایسه درنظر بگیرد)) **برای این منظور از function overloading بهره ببرید**

- double average()

این تابع میانگین مقادیر (data) نوهای لیست پیوندی را بر میگرداند ((توجه شود که اگر نوع دیتای نود از نوع عددی نبود، اگر کاراکتری بود باید براساس حروف الفبا کداسکی آنها را به عنوان مقدار عددی نود درنظر بگیرد و اگر داده از نوع رشته بود باید طول رشته را به عنوان مقدار عددی نود درنظر بگیرد)) **برای این منظور از function overloading بهره ببرید**

- 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.

علاوه بر توابع بالا که پیاده سازیشون را از تمرین قبل تا حد خوبی داشتین، این چنتا تابع کوچولو و گوگول مگولی زیر رو هم پیاده‌سازی کنین.

- int length()

این تابع کار خاصی نمیکنه فقط طول لیست پیوندی‌مان را برای ما برمیگرداند.

- void clear()

این تابع برای پاک کردن تمام دیتاهای حال حاضر در لیست پیوندیمونه.

- void reassign(Node<T>* node, T data)

این تابع پوینتری به یک نود + یک دیتا به عنوان ورودی می‌گیرد و مقدار جدید را به عنوان فیلد دیتای آن نود ست می‌کند.

- void sort()

این تابع هم که از اسمش مشخصه قراره چیکار بکنه... داده‌های لیست پیوندی از کوچ به بزرگ مرتب خواهند شد ((مجداده برای کاراکترها مقدار عددی آنها همان کداسکی آنهاست و برای رشته‌ها طول رشته))

** 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.