مقدمه:

پیچیدگی و حجم بالای اطلاعات به همراه نیاز به در دسترس بودن بهنگام آنها، همگی بیانگر لزوم ساختار بخشیدن به داده‌ها در برنامه‌های کامپیوتری هستند. به طوری که در بسیاری از برنامه‌ها، عدم وجود ساختار مناسب برای مجموعه اطلاعات به شدت بر کارایی تاثیر گذاشته، سرعت اجرای سیستم را کاهش می دهد و یا حتی کل سیستم را دچار اختلال می‌کند. لذا در اغلب مسائلی که در آن با حجم قابل توجهی داده سروکار داریم، راه‌حل‌هایی مناسب تشخیص داده می‌شوند که نه تنها درست پاسخ داده، بلکه این پاسخ صحیح را در محدودیت زمانی مورد نظر بدهند.

این چیزیست که در فاز قبلی با آن آشنا شدید. وابستگی سیستم‌ها به انجام منسجم انبوه عملیات بر روی انبوه داده‌ها در محدودیت زمانی معقول، به همراه حفظ این اطلاعات در برابر خطرات امنیتی و باگ‌های برنامه‌سازی یک مورد کاربردی است که نیاز به آن مکررا در انجام پروژه‌های متفاوت کامپیوتری در همه‌ی سطوح و زمینه‌ها، و مستقل از فضای مساله، وجود دارد. استقلال یک زیرمساله (در اینجا مدیریت انبوه داده‌ها و دسترسی به آنها) از فضایی که در آن مطرح شده و نیاز مکرر به حل دوباره این زیرمساله در کاربردهای گوناگون، حاکی از این است که اگر یک‌بار یک گروه متخصص راه‌حلی درست و جامع برای آن ارائه‌دهند، دیگر نیازی نیست در هر مساله با صرف زمان و هزینه به حل دوباره‌ی آن بپردازیم. در سیستم‌های کامپیوتری نیز با این امر مواجه هستیم و در ساختن یک سیستم (یا برنامه) استفاده از زیر سیستم‌های آماده بسیار رایج است. کتابخانه‌ها و API ها از جمله مثال‌های بارز این مساله هستند که شما نیز تا کنون با آن مواجه شده‌اید (مانند کتابخانه‌های استاندارد stdio.h، stdlib.h و...).

در این پروژه شما یک کتابخانه برای مدیریت داده‌ها پیاده‌سازی می‌کنید که همانطور که در سند مربوط به پایگاه‌داده‌ها گفته شد با وجود تفاوت‌های بسیار، مشابه یک سیستم پایگاه داده ساکن در حافظه اصلی[1] است. هدف این کتابخانه این است که به کاربرانش اجازه دهد سیستم‌ها و پروژه‌های خود را بدون نیاز به صرف زمان و هزینه روی بخش نگهداری داده‌ها پیاده‌سازی کنند.

[1] memory-resident database management system

سیستم شما باید مجموعه داده‌های متفاوت مربوط به یک برنامه را نگه‌داری کرده، ضمن برقراری اطمینان از دسترسی سریع و به موقع، امنیت داده‌ها را در برابر خطرات موجود و دسترسی نادرست تضمین کند (از طریق ایجاد امکان بازیابی). مجموعه مسائلی که در این پروژه با آن روبرو هستید را می‌توان به طور کلی به موارد زیر تقسیم کرد:

۱) **ساختار درونی سیستم و نگه‌داری داده‌ها**: کتابخانه‌ی مدیریت داده که شما طراحی می‌کنید یک سیستم جامع است. لذا داده‌هایی که در سیستم ذخیره می‌شوند فرمت‌های متفاوتی دارند (struct های متفاوت) که از ابتدا برای شما معین نیستند، رسیدگی به این مساله از جمله کارهای شما در پیاده‌سازی است.

۲) **بررسی، بهینه‌سازی و احقاق پرس‌وجو**[۲] : شما باید با سنجش پرس‌وجوها، دسترسی به رکوردها را در مدت زمان معقول امکان‌پذیر نمایید.

۳) **شاخص‌گذاری و مجموعه داده‌ها** : هر مجموعه به فرم یک لینک لیست بلند از رکوردها نگه‌داری می‌شود که داده‌های جدید در حالت کلی به انتهای این لیست اضافه می‌شوند. تحقق سرعت مورد نظر در انواع دسترسی‌ها به کمک ایندکس‌ها انجام می‌شود. پیاده‌سازی درست و حفظ لیست داده‌ها به همراه ایندکس‌های مزبور از جمله وظایف شماست.

۴) **حفظ اطلاعات در برابر خرابی** : همانطور که اشاره شد اطلاعت ما باید در صورت بروز خرابی در برنامه، قطعی برق و ... قابل بازیابی باشند. فراهم نمودن این مورد کاربرد نیز از وظایف شما در این پروژه است.

در نهایت مهم‌تر از همه، استحکام کد شما است. کتابخانه‌ی شما قرار است به دیگر برنامه‌سازها در پروژه‌هایشان کمک کرده و حجم کارشان را کاهش دهد، مسلما درستی و قابل اعتماد بودن چنین APIی تنها زمانی برآورده می‌شود که خودش بدون خطا باشد.

****در این فاز شما به پیاده‌سازی مورد اول و قسمتی از مورد دوم خواهید پرداخت.**

ساختار درونی سیستم و نگه‌داری داده‌ها

همانطور که اشاره شد، در هر پایگاه داده، تعدادی مجموعه رکورد با اسامی متفاوت وجود دارد. در بسیاری از عبارات جستجو، ما نیازمند انجام عملیات در یک مجموعه خاص هستیم که براساس اسم آن مشخص می‌شود. لذا به دنبال سریع‌ترین راهی هستیم که با استفاده از نام مجموعه به آن برسیم و طبعا نمی‌خواهیم بین تمام datasetها

[2] query evaluation

جستجو کنیم. لذا مانند فاز قبلی، از hash table استفاده می‌کنیم. هر database، یک hash table دارد که در آن اشاره‌گر به datasetها نگه‌داری می‌شود. برای رسیدگی به این موضوع، هنگام ایجاد یک dataset، ابتدا با الگوریتمی که طراحی آن به عهده‌ی خودتان هست باید مجموعه را براساس اسم آن، هش کنید و اشاره گر به آن dataset را در خانه‌ی مورد نظر در hash table اضافه کنید.

در قسمتی دیگر، همانطور که مشاهده کردید، هر واحد داده، دارای یک سری attribute (فیلد) می‌باشد. که هر attribute دارای یک کلید (key) و یک مقدار (value) است. هنگام insert کردن داده‌ی جدید، می‌بایست نام این attribute ها مشخص باشد تا بعدا بتوان بر روی آن‌ها جستجو انجام داد. چون در این جا نیز باز یک مکانیزم جفت کلید و مقدار داریم (یعنی دسترسی به value ها براساس key شان) نیازمند داشتن یک hash table برای نگه‌داری attribute های هر داده هستیم. لذا می‌بایست با یک تابع درهم‌سازی، key ها را هش کرده و value ها در خانه‌ی مورد نظر ذخیره کنیم.

تصادم

در فاز قبلی با توابع درهم‌سازی آشنا شدید و دیدید که مفهومی به نام تصادم وجود دارد که باعث می‌شود دو کلید متفاوت به یک مقدار مساوی تبدیل شوند. در این فاز می‌خواهیم این مشکل را حل کنیم. برای حل این موضوع راه‌حل‌های متعددی وجود دارد که ما در این درس صرفا یک راه حل ساده‌ی آن به نام separate chaining را پیاده‌سازی می‌کنیم.

ایده‌ی این راه‌حل این هست که در خانه‌ی مربوط به value ها، به جای اینکه یک مقدار را نگه‌داری کنیم، linked listای از مقادیری که hash شده ی کلید آنها یکسان است را نگه داری می کنیم. هر node در این linked list دارای یک مقدار کلید و یک مقدار (یا یک اشاره‌گر در صورتی که حوزه‌ی بحث، واحدهای داده هستند نه attribute ها) است. برای بازیابی یک داده، بر روی linked list حرکت کرده و node ی که کلیدش با کلید داده شده برابر بود را برمی‌گردانیم.

البته توجه کنید که تنها زمانی که تعداد تصادم‌ها کم باشد، حرکت روی این linked list هزینه‌ی چندانی نخواهد داشت ولی اگر تعداد تصادم‌ها زیاد باشد، فرقی با حرکت روی آرایه برای پیدا کردن داده نخواهد داشت.

بررسی و احقاق پرس‌وجو

تعامل با دیتابیس و تغییر داده‌ها از طریق دستورهایی صورت می‌پذیرد که پرس و جو (query) نامیده می‌شوند. دستوراتی که در این فاز شما پیاده‌سازی می‌کنید شامل ایجاد و حذف دیتابیس، ایجاد و حذف مجموعه‌داده، انتخاب

(که داده‌هایی را که با معیار مشخص شده در پرس‌وجو همخوانی دارند برمی‌گرداند) و حذف داده است. توضیحات این دستورها به طور مفصل در کامنت‌های هدرفایل آمده است.

زبان‌های پرس‌وجو[3] (پرسش) زبان‌های برنامه‌نویسی برای اعمال پرسش (درخواست) بر پایگاه‌های داده است. زبان‌های پرس‌وجو امکان دست‌یابی و تغییر رکوردها یا مجموعه‌رکوردها را فراهم می‌کنند. شرح ساختار زبان پرس‌وجویی که در این پروژه خواهیم داشت در ادامه می‌آید.(این ساختار زبان در فاز بعدی کامل‌تر می‌شود)

در ادامه با تعدادی از دستورات این زبان آشنا می‌شویم

```
DATASET:collection_name field1:type1 field2:type2 field3:type3 …
```

دستور اول دستور ساختن یک مجموعه داده است. با دستور DATASET یک مجموعه‌رکورد در پایگاه داده ساخته می‌شود. این دستور از دو بخش تشکیل شده است. بخش اول نام مجموعه مورد نظر است و بخش دوم لیست نام فیلدها و نوع شان را مشخص می‌کند. جدول زیر انواع داده‌ی ابتدایی است که پایگاه داده‌ی شما ساپورت می‌کند.

DATA TYPE	***DESCRIPTION***
STRING	*Character string. Variable length. Maximum length 50*
BOOLEAN	*Stores True or False*
INTEGER	*Integer numerical (no decimal).*

برای مثال به دستور زیر توجه کنید :

DATASET:students name:STRING sid:STRING graduated:BOOLEAN year:INTEGER gpa:INTEGER

در این دستور مجموعه رکوردی با نام students ایجاد می‌کنیم که شامل فیلدهای "نام" از نوع رشته، "شماره‌ی دانشجویی" از نوع رشته، "سال ورود دانشجو" از نوع صحیح، "یک متغیر منطقی که فارغ‌التحصیل بودن یا نبودن دانشجو را مشخص کند" و "معدل دانشجو" از نوع صحیح است.

[3] Query Languages

دستور بعدی، دستور insert است که به وسیله‌ی آن می‌توان داده به یک مجموعه اضافه کرد. در این دستور نیز

```
INSERT_INTO:collection_name  field1:value1 field2:value2 …
```

ابتدا نام مجموعه و سپس نام هر فیلد به همراه مقدار آن می‌آید.

مثال:

INSERT_INTO:students name:“Ali Alizadeh” sid:“810193123” graduated:FALSE year:1393 gpa:20

```
SELECT:collection_name condition1 condition2 …
```

یکی از مهم‌ترین دستورات، دستور SELECT است. که از طریق آن به رکوردها دسترسی پیدا می‌کنیم.

دستور select برای دست‌یابی به داده‌ها به کار می‌رود. در این دستور معیار برای انتخاب داده‌ها برآورده شدن همه شرط‌های زیر است. (در این فاز صرفا دستور برابری را پیاده‌سازی می‌کنید، در فاز بعدی سایر عبارات را نیز اضافه می کنیم):

EXAMPLE: SELECT:students gpa:20 graduated:TRUE

مجموعه‌ای از شرط‌هایی که می‌تواند در دستور SELECT اعمال شود در جدول زیر آمده‌است.

OPERATOR	DESCRIPTION
:	Equal
<>	Not equal.
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

دو دستور DELETE و UPDATE نیز برای تغییر در مجموعه‌ها به کار می‌روند. ساختار این دو دستور مشابه دستور SELECT است. DELETE داده‌های واجد شرایط را از مجموعه حذف می‌کند و UPDATE تغییرات مشخص شده را بر این داده‌ها اعمال می‌کند. که در ادامه ساختار آن را مشاهده می‌کنید:

```
DELETE:collection_name condition1 condition2 condition3 …
```

```
UPDATE:collection_name condtion1 condition2 condition3 …
      field1:value1 field2:value2
```

API شما دستورات فوق را در قالب structهایی که در هدر تعریف شده می‌پذیرد. این structها دقیقا نمایش دهنده‌ی زبان توضیح داده‌شده است. شرح کامل آن‌ها در هدرفایل مربوطه آمده است.

آن چه شما پیاده‌سازی می‌کنید:

در این فاز شما با پیاده‌سازی توابع و struct مشخص شده در هدر فایل با توجه به توضیحات آمده و کامنت‌های هدرفایل بخش مهمی از کتابخانه‌ی خود را پیاده‌سازی خواهید کرد. کد شما و توابع پیاده‌سازی شده باید کاملا مطابق با این هدرفایل باشند. مسلما بخش بندی و ساختار درونی کتابخانه مستقل از هدر، به طراحی شما بستگی دارد و شما باید ساختار درونی درستی را (با تقسیم بندی صحیح توابع و تعریف struct های جدید) پیاده‌سازی کنید.

کد پیاده‌سازی شده شما باید تا حد ممکن نسبت به تغییرات منعطف بوده و بدون خطا باشد. مانند تمرین‌های متعدد گذشته در تحویل این پروژه نیز برخورد با هرگونه runtime error، کرش کردن یا در لوپ افتادن، دارای پنالتی قابل توجه خواهد بود. لذا با دقت و درست کد بزنید. تست صحیح، مرحله به مرحله و همگام با تولیدکد بسیار مهم بوده و کار شما را هم آسان‌تر خواهد کرد. ارائه‌ی این تست‌ها، توجیه درست تقسیم و اصولی پیاده‌سازی کردن پروژه بخشی از نمره شما خواهد بود.

تحویل برنامه:

شما باید یک فایل فشرده zip. آپلود کنید که نام این فایل شامل نام، نام خانوادگی و شماره دانشجویی شما باشد (مثلا name_family_810193123.zip). که در این فایل فایل‌های هدر و سورس خود را قرار می‌دهید. از آپلود کردن فایل‌های اضافی جدا خودداری فرمایید (پنالتی دارد). کسانی که این فرمت را رعایت نکنند به مشکل برخواهند خورد.

سیستم نمره‌دهی:

نمره‌دهی این فاز به صورت حضوری خواهد بود. در تحویل حضوری فایل‌های شما به همراه هدرفایلی که در اختیارتان گذاشته‌ایم، در برنامه‌های تست ما استفاده می‌شوند (مشابه کتابخانه هایی که شما استفاده می‌کنید) و پاسخ به صورت اتوماتیک چک شده این تست‌ها ۶۰٪ نمره‌ی شما را تشکیل می‌دهند. ۴۰٪ درصد دیگر نمره‌ی شما:

- رعایت اصول کدینگ (تقسیم‌بندی درست برنامه، کاهش code-duplication و ...) : ۱۰
- مدیریت درست حافظه (free کردن، عدم وجود memory leak و ...) : ۱۰
- پیاده‌سازی و استفاده‌ی صحیح linked-listها: ۱۰
- پیاده‌سازی و استفاده‌ی صحیح از جدول درهم‌سازی: ۱۰

در آخر دقت کنید که مواردی مانند لینک لیست‌ها و جداول درهم‌سازی و... نقش مرکزی در پروژه دارند. و عدم استفاده از آن‌ها به کم شدن نمره‌ی بخش مربوطه ختم نشده و پنالتی خواهد داشت.

نکات پایانی:

- **این فاز بخش مهمی از پروژه‌ی پایانی شما خواهد بود و حجم کار شما نسبت به تمرین‌های قبلی بسیار بیشتر است. لذا توصیه می‌شود زودتر انجام آن را شروع کنید و با دقت انجام دهید.**
- **برنامه‌ی شما در یک محیط استاندارد تحویل گرفته می‌شود. پس باید از توابع استاندارد C استفاده کنید. شرط استاندارد بودن، وجود آن در یکی از کتابخانه‌های بخش C Library در این آدرس است.**

- **دقت کنید در هنگام نمره‌دهی غیر حضوری، در صورت برخورد با runtime error یا در لوپ افتادن برنامه‌ی شما در هرکدام از تست‌ها، 40 درصد از نمره‌ی کل را از دست خواهید داد.**
- **این پروژه یک کار تک‌نفره است!**
- **Comment نویسی *درست* در کد الزامی است.**
- **تمیز بودن کد شما اهمیت ویژه‌ای دارد. عدم رعایت فاصله از سر خط[۴] در کدنویسی و نام‌گذاری‌های نامناسب تا ۱۵٪ از نمره‌ی نهایی شما خواهد کاست.**
- **در صورت مشاهده‌ی هر گونه تشابه بین برنامه‌ی دو یا چند نفر، نمره‌ی تمامی افراد شرکت‌کننده در تقلب صفر خواهدشد.**
- **پروژه‌ی شما حتماً باید به زبان C (و نه C++) باشد. یعنی حق استفاده از هیچ کدام از کتابخانه‌های استاندارد C++ (مانند iostream،vector و ...) را ندارید. در صورت رعایت نکردن این مسئله نمره‌ی صفر برای شما لحاظ می‌گردد.**
- **در صورت عدم تسلط به برنامه‌ی خود در زمان تحویل، نمره‌ی صفر خواهید گرفت.**
- **در صورت استفاده از دستور goto، متغیرهای گلوبال و دستور system() نمره‌ی شما صفر خواهد شد.**

خوراک بیشتر:

در راستای درک بهتر الگوریتم‌های مطرح شده، کسب اطلاعات بیشتر و اغنا(!) اطلاعات خود لینک‌های زیر وجود دارند. البته منابع بسیار خوب دیگری هم در وب وجود دارد که با جستجو می‌توانید پاسخ سوالات خود را در آن‌ها بیابید.

http://www.pcmag.com/encyclopedia/term/37856/api

https://en.wikipedia.org/wiki/Database

https://en.wikipedia.org/wiki/In-memory_database

http://www.tutorialspoint.com/dbms/dbms_data_models.htm

http://www.tutorialspoint.com/dbms/dbms_hashing.htm

/http://www.tutorialspoint.com/dbms

شاد باشید و موفق

---

[4] indentation