pizza

Data Structure Project

PIZZA FINDER

Submission deadline:

1402/11/14

مقدمه:

آیا تا به حال برای شما اتفاق افتاده است که هوس پیتزا کرده باشین اما ندانید نزدیکترین پیتزا فروشی کجا قرار دارد؟ یا اینکه قصد رفتن به پیتزا فروشی خاصی را داشته باشید؟ درخت‌های چند بعدی (KD-Tree)از جمله ساختارهایی هستند که به ما کمک فراوانی در این زمینه می‌کنند. در پروژه پایانی این درس شما باید منطق برنامه ای را پیاده سازی کنید که این نیاز را برطرف می‌کند. برنامه شما تحت کنسول اجرا می‌شود. همچنین این برنامه باید تا حد امکان بهینه نوشته شده باشد و نمره شما بر اساس درستی کارکرد توابع و بهینگی آنها داده خواهد شد.

افزودن محله:

با دستور **Add-N** برنامه شما باید مختصات یک مستطیل (که بیانگر محله می باشد) و همچنین نام محله را از کاربر دریافت کند. (ممکن است محله ها اشتراک داشته باشند)

افزودن پیتزا فروشی اصلی (شعبه اصلی):

با دستور **Add-P** برنامه شما باید مختصات پیتزا فروشی را از کاربر دریافت کند و پیتزا فروشی را اضافه کند. (در صورت وجود پیتزا فروشی دیگر در آن نقطه، برنامه شما باید پیغام خطای مناسب را چاپ کند).

هر پیتزا فروشی دارای موارد زیر است:

- مختصات پیتزا فروشی (برای سادگی کار، فرض کنید پیتزا فروشی ها نقاطی در صفحه هستند)
- نام پیتزا فروشی

افزودن شعبه پیتزا فروشی:

با دستور **Add-Br** برنامه شما باید مختصات شعبه پیتزا فروشی را از کاربر دریافت کند و پیتزا فروشی را اضافه کند (در صورت وجود پیتزا فروشی دیگر در آن نقطه، برنامه شما باید پیغام خطای مناسب را چاپ کند).

هر شعبه دارای موارد زیر است:

- مختصات پیتزا فروشی (برای سادگی کار، فرض کنید پیتزا فروشی ها نقاطی در صفحه هستند)
- نام پیتزا فروشی
- نام شعبه اصلی

حذف شعبه پیتزا فروشی:

با دستور **Del-Br** برنامه شما با دریافت مختصات یک نقطه، پیتزا فروشی موجود در آن نقطه را در صورت وجود حذف می کند. در صورت عدم وجود پیتزا فروشی و یا پیتزا فروشی اصلی بودن در آن نقطه نیز پیغام مناسب را چاپ کنید.

لیست تمامی پیتزا فروشی های یک محله:

با دستور **List-P** برنامه شما باید نام یک محله را از کاربر دریافت کند و مشخصات پیتزا فروشی های موجود در آن محله را چاپ کند.

مختصات تمامی شعب یک پیتزا فروشی:

با دستور **List-Brs** برنامه شما باید با دریافت نام یک پیتزا فروشی، مختصات تمامی شعب آن پیتزا فروشی را چاپ کند.

نزدیک‌ترین پیتزا فروشی:

با دستور **Near-P** برنامه شما باید یک مختصات از کاربر دریافت کند و مشخصات نزدیک ترین پیتزا فروشی را چاپ کند.

نزدیک‌ترین شعبه پیتزا فروشی:

با دستور **Near-Br** برنامه شما باید با دریافت یک مختصات و نام یک پیتزا فروشی، مختصات نزدیکترین شعبه از آن پیتزا فروشی را چاپ کند.

تمامی پیتزا فروشی های قابل دسترس:

با دستور **Avail-P** برنامه باید عدد **R** و همچنین یک مختصات از کاربر دریافت کند و مشخصات تمامی پیتزا فروشی هایی که از مختصات داده شده به شعاع **R** موجود هستند را چاپ کند.

پر شعبه‌ترین پیتزا فروشی:

با دستور **Most-Brs** برنامه شما باید نام پیتزا فروشی که بیشترین تعداد شعبه دارد را چاپ کند. برای پیاده سازی این قسمت ابتدا یک لیست از تعداد شعبات همه پیتزا فروشی های اصلی ایجاد کرده و سپس روی آن یک سورت انجام داده و بیشترین مقدار را خروجی دهید.

عقب کشیدن زمان (امتیازی):

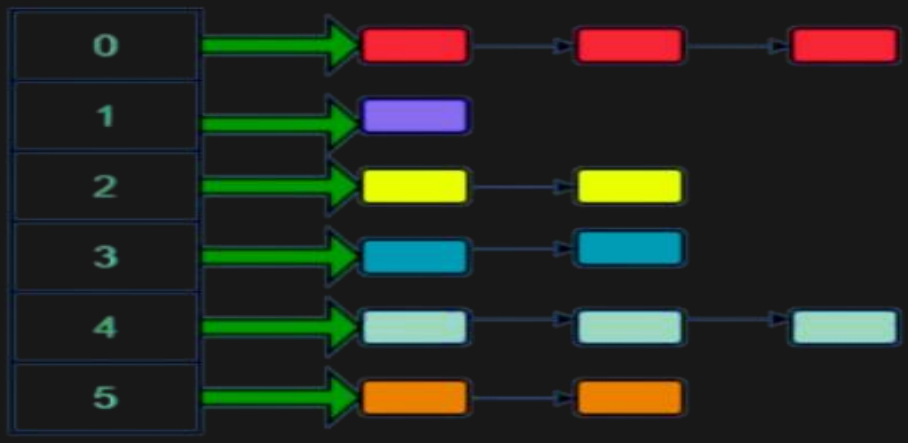
با دستور **Undo** برنامه شما باید یک عدد **P** از کاربر دریافت کند و برنامه را به زمانی برگرداند که **P** امین دستور اجرا شد(به عنوان مثال اگر تا الان شش دستور انجام شده است و قصد داریم به زمان اجرای دستور سوم برویم، اثرات دستور چهارم و پنجم و ششم حذف خواهد شد). توجه کنید که در هر زمان ممکن است چند دستور اجرا شود که با **&&** از هم جدا می‌شوند و باید بتوان بین تاثیر دستور هر زمان نیز جابجا شد. برای مثال اگر در یه زمان دستورات به شکل زیر وارد شده باشد:

```
Del-Br [BRANCH_NAME] && Add-Br [BRANCH_NAME] && Add-P [PIZZA_NAME]
```

باید بتوان تاثیر دستور اول یا دوم یا سوم را بر حسب خواسته کاربر دید .برای پیاده سازی این قسمت میتوانید از ساختار **Hash** با **chaining** (استفاده از لینکد لیست) استفاده کنید.

نکات مهم:

۱- از آنجایی که تمامی شعبات و پیتزا فروشی‌ها بر اساس نام آن‌ها شناسایی می‌شوند و جست و جو آن‌ها بر این اساس است، پس باید ساختار بهینه‌ای برای این جست و جو طراحی کنید که پیچیدگی زمانی آن بهینه باشد (اگر بهینگی رعایت نشود مقدار زیادی از نمره مربوطه کسر خواهد شد). شما باید برای این منظور از ساختار **Hash** استفاده کنید.



۲- گروه های پروژه می‌تواند حداکثر ۲ نفره باشد. همچنین ارئه پروژه بصورت حضوری است و در زمان تحویل ممکن است از شما خواسته شود تغییراتی در کد اعمال کنید. بنابراین نوشتن کد تمیز و قابل فهم اهمیت زیادی دارد.

۳- پیاده سازی ساختمان داده **KD-Tree** و رعایت اصول شئ گرایی و چند فایلی بودن الزامیست.

۴- قسمت امتیازی تنها در صورتی که موارد اجباری به طور کامل و بودن نقص پیادی سازی شده باشند لحاظ خواهد شد.

۵- تمامی کد های با استفاده از نرم افزار مشابه گیر **MOSS** بررسی خواهد شد. هر دو پروژه‌ای با یکدیگر نباید بیش از ۱۰ درصد شباهت داشته باشد. در صورت یافتن تشابه در کد ها، نمره پروژه صفر در نظر گرفته خواهد شد.

۶- برای انجام پروژه باید از زبان **C++** استفاده کنید. استفاده از کتابخانه های آماده الگوریتمی و ساختمان داده های آماده مجاز نیست.

موفق باشید(: