به نام خدا

تمرین عملی سری سوم درس ساختمان داده – بخش دوم

نکته: این تمرین شامل یک پروژه تک نفره که صورت آن در زیر بیان شده می‌باشد. این پروژه ۴۰ نمره دارد. یک سؤال ساده نیز که ۱۰ نمره دیگر دارد در پایان جزوه وجود دارد. مجموع نمرات این تمرین ۵۰ است.

نکته: پس از ایجاد یک ریپوزیتوری در گیتهاب توسط حل التمرین عملی درس، شروع به انجام مرحله به مرحله پروژه کنید، این مراحل را در فایل `readme.md` توضیح دهید، این توضیحات باید در حالتی خیلی جزئی باشند. نمره این بخش ۱۰ است و درصورت عدم وجود توضیحات حل التمرین **مجاز به صفر** کردن نمره کل پروژه می‌باشد.

نکته: سعی کنید `commit` های مکرر داشته باشید به صورتی که بعد از انجام هر بخش یک `commit` انجام شده باشد، این کار مشخص میکند پروژه در چند روز و در چند مرحله توسط شما انجام شده. این کامیت‌ها را سعی کنید بلافاصله در ریپوزیتوری `push` کنید.

نکته: قبل از شروع تمرین انتظار می‌رود ساختمان داده‌های **آرایه پویا، آرایه ایستا، صف، پشته، لینک لیست یک‌طرفه** و **دوطرفه، درخت جست‌وجوی دودویی، درخت Red-Black، درخت AVL** و **درخت Heap** تدریس شده باشد، با توجه به این موضوع لطفاً سعی کنید از ساختمان داده‌هایی مانند درخت ترای، جدول درهم سازی و.. استفاده **نکنید**.

صورت پروژه از قرار زیر است، پروژه به گونه‌ای تنظیم شده که ظرف یک هفته قابل انجام باشد:

اورژانس بیمارستان شهید محمدی شهر بندرعباس از شما برنامه‌ای می‌خواهد تا برای مدیریت ارسال آمبولانس‌هایش برای انتقال بیماران اورژانسی به بخش اورژانس استفاده کند. در این برنامه یک مسئول سیستم وجود دارد که از پنل کاربری می‌تواند در ابتدا تعدادی آمبولانس در سیستم تعریف کند، هر آمبولانس صرفاً یک نام دارد، که با این نام یکتا در پنل شناخته می‌شود، همچنین برای هر آمبولانس یک سرعت تعیین می‌شود، در نتیجه از برنامه شما انتظار می‌رود برای بیماران با وضعیت وخیم از آمبولانس‌های پرسرعت خود استفاده کند. مسئول سیستم بعد از تعریف تعدادی آمبولانس‌ها می‌تواند پس از دریافت هر تماس ورودی یک بیمار وارد صف انتظار آمبولانس کند، برنامه شما در اولین فرصت باید یکی از آمبولانس‌های آزاد را به محل اعزام کند، برای هر بیمار یک آدرس (اختیاری)، شماره تماس (اختیاری)، نام و وخامت بیمار که یک عدد مابین ۱ تا ۱۰۰ است در نظر بگیرید. تعداد بیمارانی که به صورت همزمان ممکن است در صف انتظار باشند حداکثر می‌تواند ۲۰۰ نفر باشد.

اولویت هر بیمار پس از هر ده ثانیه یک واحد بیشتر می‌شود، یعنی اگر یک بیمار با اولویت ۱۰ در برنامه ثبت شود، سپس ده دقیقه بعد اولویت او ۷۰ است و اگر بیمار جدیدی با اولویت ۶۰ وارد شود، این بیمار در اولویت اول قرار دارد. (راهنمایی، از آنجایی که میزان رشد اولویت‌ها خطی و بین همه بیماران برابر است، در نتیجه ترتیب بیماران بهم نخواهد خورد)

بخش‌های پنل کاربری:

۱. کلید افزودن آمبولانس، برای هر آمبولانس ورودی‌های **نام**، **سرعت** دریافت کنید.

۲. لیست آمبولانس‌ها، روبروی هر آمبولانس **سرعت**، **وضعیت سرویس دهی** (آزاد یا ماموریت)، **زمان گذشته از آخرین اعزامی** (فقط در صورتی نمایش دهید که وضعیت سرویس دهی برابر ماموریت باشد، در این حالت مشخص کنید از زمان اعزام آمبولانس چه مدت زمان میگذرد)، علاوه بر آن مسئول می‌تواند اطلاعات بیماری که آمبولانس برای آن اعزام شده را مشاهده کند.

۳. از لیست آمبولانس‌ها مسئول بخش با کلیک برروی هر آمبولانسی که وضعیت آن ماموریت باشد می‌تولند به برنامه اعلام کند که آمبولانس از ماموریت بازگشته و آماده ماموریت بعدی است، در حالتی که بیماری در انتظار آمبولانس باشد برنامه باید فوراً پیغامی که اطلاعات بیمار بعدی را نمایش می‌دهد به کاربر نشان

دهد، پس از تأیید مسئول بخش، وضعیت آمبولانس به حالت ماموریت در می‌آید.

۴. گزینه افزودن بیمار، در این هنگام به دلیل نیاز به تسریع کار ابتدا مسئول بخش نام بیمار و اولویت او را وارد می‌کند، سپس سیستم باید بلافاصله در صورت خالی بودن آمبولانسی اعزام کند (البته همانطور که در مورد قبل گفته شد ابتدا پیغام نمایش داده شود و در صورت تأیید مسئول آمبولانس اعزام شده در نظر گرفته می‌شود). پس از اعزام آمبولانس اکنون مسئول بخش می‌تواند با آسودگی خاطر شماره تماس و آدرس محل را در صورت تمایل وارد کند. (فرض کنید آمبولانس می‌داند کجا برود! مثلاً از روی موقعیت جی پی اس تماس)

۵. اولین بیماری که در انتظار آمبولانس است (به ترتیب اولویت)، و البته ممکن است هیچ بیماری منتظر نباشد. نام بیمار و وضعیت وخامت را نمایش دهید، در صورت کلیک برروی این فیلد، اطلاعات کامل بیمار باز شود.

سؤال ۱:

یک درخت AVL بنویسید شامل متدهای insert و find و show.

```python
class AVL:
    def insert(self, key, data):
        # insert the data with it's key to the tree
        pass

    def find(self, key):
        # returns the data of the key
        pass

    def show(self):
        # returns an string presenting the tree and balance factor of each node.
        pass
```

موفق باشید.