بسمه تعالی

درس: ساختمان های داده ها　　دانشگاه فردوسی مشهد　　مدرس: غیاثی شیرازی

تمرین HeapSort:

در این تمرین شما می‌بایست بدنه‌ی توابع زیر را در فایل Heap.h :

```
    void initialize(T* data, int n)
    {
        // Write your code here
    }

    virtual T deleteMax()
    {
        // Write your code here
    }

    virtual void insert(const T& data)
    {
         // Write your code here
    }
```

و بدنه تابع sort در فایل HeapSort.h را تکمیل کنید:

```
virtual void sort(T* data, int n)
    {
        // Write your code here
    }
```

تابع initialize دو پارامتر دریافت می‌کند:

۱. data: آرایه متناظر با درخت heap است. توجه داشته باشید که این درخت یک درخت کامل است. به این معنا که اگر گره ها را شماره گذاری کنیم شماره هر گره متناظر با شماره همان گره در یک درخت پر با عمق برابر آن است. در نتیجه اگر شماره گذاری از ۱ شروع شود فرزند چپ گره i ام گره 2*i ام و فرزند راست گره 1+2*i ام است. همچنین در این درخت امکان ندارد که فرزند راستی وجود داشته باشد در حالی که فرزند چپ وجود ندارد. (جهت اطلاعات بیشتر به مطالب درس مراجعه نمایید)
۲. n: اندازه درخت یا همان تعداد گره ها.

در این تابع قصد داریم درخت (آرایه) را به گونه ای تغییر دهیم تا تبدیل به یک درخت Max Heap شود. به عبارتی هر گره بزرگترین گره زیردرخت خود باشد.

در تابع deleteMax باید بزرگترین گره درخت که همان ریشه است حذف شود و باتوجه به مطالب درس باقی گره ها به گونه ای تغییر کنند که درخت همچنان یک Max Heap بماند. (یادآوری: پس از حذف ریشه، آخرین گره را در ریشه قرار داده و آن را به جای درست خود منتقل می‌کنیم)

در تابع insert یک گره در آخر درخت (شماره بعد از آخرین گره) قرار می‌گیرد و سپس با مقایسه با گره پدر و جابجایی های مکرر به مکان صحیح خود منتقل می‌شود. این تابع یک پارامتر دریافت می‌کند که همان داده جدید است.

تابع sort در HeapSort که دو پارامتر دریافت می‌کند:

۱. data: که آرایه متناظر با یک Max Heap است.
۲. n: که اندازه درخت Max Heap است.

این تابع آرایه متناظر با Max Heap را تبدیل به یک آرایه صعودی می‌کند. (یادآوری: عملکرد این تابع به این گونه است که در هر مرحله ریشه که بزرگترین عضو است را حذف کرده و به آخر آرایه منتقل می‌کند. در نتیجه آخرین عضو را به ریشه آورده و درخت Max Heap را دوباره تنظیم می‌کند. این عمل آن قدر تکرار می‌شود تا به آرایه مرتب شده برسیم. جهت مشاهده مثال می‌توانید به مطالب درس مراجه کنید)

ما برای پیاده سازی این ساختمان داده یک متود خصوصی به نام heapify نوشتیم که اندیس ریشه یک زیردرخت را دریافت می-کند و با فرض اینکه زیردرخت‌های چپ و راست خود Heap هستند، کل زیردرخت را Heap می‌کند. اما دانشجویان مختارند این تابع را به نحوی که تمایل دارند پیاده‌سازی کنند و به همین دلیل، در تعریف تمرین، تابع heapify مشاهده نمی‌شود.

در این تمرین نحوه ارزیابی به شرح زیر است:

۱. testSwapCounter: هدف از این تست بررسی بهینه و درست بودن کد شماست. در این تست تعداد دستورات انتصاب که نمایانگر تعداد جابجایی عناصر در درخت است شماره شده و در صورت انجام جابجایی های زیاد یا انتصاب های غیر ضروری از این تست نمره ای دریافت نخواهید کرد. همچنین ذکر این نکته لازم است که این تست پیش نیاز بقیه تست ها می‌باشد و پاس نشدن آن به معنا پیاده سازی نادرست درخت Max Heap بوده و از بقیه تست ها نیز نمره‌ای دریافت نخواهید کرد. این تست به تنهایی ۱۰٪ از نمره تمرین را به خود اختصاص می‌دهد.
۲. testInit: در این تست تابع initialize تست می‌شود. به این صورت که روی تعدادی آرایه این تابع صدازده شده و انتظار می‌رود خروجی یک درخت Max Heap باشد. از آنجا که در تست های دیگر از خروجی این تابع که درخت Max Heap مورد نظر است استفاده می‌شود، طبیعی است که در صورت پیاده سازی نادرست این تابع از دیگر تست ها نمره ای دریافت نکنید. این تست ۲۰٪ نمره این تمرین را به خود اختصاص می‌دهد.
۳. testInsert: در این تست تعدادی گره به درخت اضافه می‌شود و بررسی می‌شود که به درستی در جای مناسب خود قرار گرفته باشند. وجود این تابع برای عملکرد مناسب HeapSort ضروری نیست، اما از آنجایی که عملیات مهمی در ساختمان داده Heap است ۲۰٪ از نمره این تمرین را به خود اختصاص داده است.
۴. testDeleting: در این تست تابع deleteMax تست می‌شود. در این تست تعدادی گره از درخت Max Heap حذف می‌شوند. از آنجا که عملکرد HeapSort بر مبنا حذف ریشه یا به عبارتی همان تابع deleteMax است، طبیعی است که در صورت پیاده سازی نادرست این تابع از testSort نیز نمره ای دریافت نکنید. این تست ۲۰٪ از نمره این تمرین را به خود اختصاص می‌دهد.
۵. testSort: در این تست درخت Max Heap ای به تابع داده شده و انتظار می‌رود تا آرایه صعودی آن را دریافت کنیم. این تست ۳۰٪ از نمره این تمرین را به خود اختصاص می‌دهد.

برای انجام این تمرین کارهای زیر را انجام دهید:

۱- ابتدا در این پوشه فایل info.txt را با مشخصات خود پر کنید.
۲- سپس یک پروژه Visual Studio C++ بسازید و فایل های پوشه های src و test را به آن اضافه کنید.
۳- تمرین را انجام دهید و بخش‌های ناقص کد را تکمیل کنید و از درستی پیاده سازی خود مطمئن شوید.
۴- از پوشه src همه فایل های اضافی که به دلیل کامپایل برنامه بوجود آمده اند را پاک نمایید. (پوشه Debug و فایل با پسوند sdf و نیز پوشه مخفی vs را حتما پاک کنید زیرا حجم زیادی می گیرند).
۵- محتویات کل پوشه را به صورت یک فایل zip در آورید.
۶- مطمئن شوید که وقتی فایل zip را باز می کنید پوشه های src, img و test و همچنین فایل info.txt را می بینید.
۷- نام این فایل zip را به «شماره تمرین-شماره دانشجویی» تغییر دهید (مثل: 123456789- HeapSort.zip)
۸- دقت کنید که پسوند فایل شما حتما zip باشد.
۹- اگر حجم فایل بالای یک مگابایت باشد، حتما در پاک کردن محتویات بوجود آمده هنگام کامپایل (مرحله ۴) اشتباه کرده‌اید.
۱۰- ابتدا این فایل را به سیستم «سپهر» ایمیل کنید تا از نحوه عملکر برنامه خود بر روی تست های تکمیلی آگاه شوید.
۱۱- اشکالاتی را که سیستم «سپهر» مشخص کرده است برطرف نمایید و مجددا تمرین را به سیستم «سپهر» تحویل دهید.
۱۲- مرحله قبل را آن قدر ادامه دهید که از صحت عملکرد برنامه خود اطمینان حاصل نمایید.
۱۳- پس از اطمینان از دریافت نمره کامل، همان فایل ارسالی را از طریق سیستم vu تحویل دهید.

با آرزوی موفقیت