بسمه تعالی

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

تمرین BinaryTreeIterationWithoutParentNodes:

در این تمرین قرار است درخت دودویی پیاده سازی شود تا امکان پیمایش به انواع گوناگونی که در درس آمده است را بر روی درخت داشته باشد. جهت تکمیل این تمرین، دو فایل با نام های BinaryTree.h و InternalBinaryTreeNode.h در اختیار شما قرار گرفته است. فایل InternalBinaryTreeNode.h شامل کلاسی است که گره های مربوط به درخت را می‌سازد. پس ابتدا لازم است تا این فایل را مطالعه نموده و از نحوه ذخیره‌سازی اطلاعات گره‌های درخت مطلع شوید. نکته بسیار مهم در این تمرین این است که گره‌های داخلی درخت تنها شامل فرزند چپ و راست (و البته داده ذخیره شده در گره) هستند و اشاره‌گری به گره پدر وجود ندارد. پس از بررسی این موضوع و مطالعه گره‌های داخلی، به کمک کلاس مربوط به گره داخلی توابع مورد نظر را در BinaryTree.h پیاده‌سازی نمایید و تمرین را تکمیل کنید. برای پیاده‌سازی این توابع درون هر پیمایشگر یک پشته از گره‌های پدر است که لازم است دانشجویان از آن برای نگه‌داری اجداد گره فعلی استفاده کنند و هر زمان که نیاز به بازگشت به گره پدر بود از آن کمک بگیرند. در این فایل لازم است توابع زیر به درستی کامل گردند.

۱. این تابع گره مربوط به پدر تمام dummy root ها را برمی‌گرداند. همان طور که درس هم آمده، این گره اولین گره در درخت است که هیچ پدری ندارد و یکی از فرزندان آن (یا به عبارتی فرزند فزندان آن) درخت اصلی است. این گره‌ها برای شناسایی نقطه پایان پیمایش استفاده می‌شوند.

```
    virtual BinaryTreeNode  getHeaderRootNode(){
        // Write your code here
    }
```

توجه کنید که قبل از پیاده‌سازی این تابع لازم است در این قسمت گره های مربوط به dummy root ها را ایجاد کنید.

```
protected:
    // Write your code here (Add appropriate attributes for dummy roots)

    int     mNodeDisplayWidth;
    int     mSize;
```

همچنین در این قسمت شی مربوط به گره ها را بسازید و اتصالات بین گره های dummy root را برقرار نمایید.

```
public:
    BinaryTree(void){
        mNodeDisplayWidth = 2;
        mSize = 0;
        // Write your code here
    }
```

*** توصیه اکید می‌شود قبل از پیاده‌سازی این قسمت مطالب مربوط به dummy root ها را از درس مطالعه نمایید تا این قسمت را به درستی متوجه شوید و به یک نتیجه صحیح برسید.

۲. در این تابع اطلاعاتی به عنوان پارامتر دریافت می‌شود و در گره ریشه‌ای که ساخته می‌شود، قرار می‌گیرد. همان طور که اطلاع دارید در دنیای کامپیوتر وقتی با درخت ها سر و کار داریم، داشتن گره ریشه به معنا داشتن تمام درخت است و به کمک آن می‌توانید تمام درخت را پیمایش کنید و به تمامی اطلاعات آن دسترسی داشته باشید. پس این تابع به نوعی سازنده درخت است و به کمک این تابع است که اولین گره را به درخت اضافه می‌کنیم و شروع به ساختن آن می‌نماییم. پارامتر های این تابع به صورت زیر است:

data: که شامل اطلاعات مربوط به گره ریشه است.

```
    virtual void    insertRootNode(T data){
        // Write your code here
    }
```

۳. در این دو تابع، گره و اطلاعاتی را به عنوان ورودی دریافت می‌کنید و بر اساس نوع تابع، آن اطلاعات را در فرزند چپ یا راست گره دریافت شده، قرار می‌دهید. پارامتر های این توابع به صورت زیر است:

parentNode: این پارامتر گره مورد نظر را می‌دهد که بر اساس نوع تابع اطلاعات در گره جدید به عنوان فرزند آن قرار می‌گیرد.

data: این پارامتر اطلاعات مورد نظر است که باید در گرهی جدید به عنوان فرزند گره پدر ذکر شده اضافه شود.

```
    // error if a left child already exists.
    virtual void    insertLeftChild(const BinaryTreeNode& parentNode, T data){
        // Write your code here
    }

    // error if a right child already exists.
    virtual void insertRightChild(const BinaryTreeNode& parentNode, T data){
        // Write your code here
    }
```

* در این دو تابع، در صورتی که گره ارسال شده به عنوان پدر از قبل فرزندی داشت و امکان اضافه شدن فرزند جدید در آن قسمت را نداشت، exception ای ارسال کنید تا از ادامه برنامه جلوگیری نماید.

۴. در این تابع باید گرهی را از درخت حذف نمایید. پس برای این کار، تابع به همراه گره مورد نظر و گره پدر آن صدا زده می‌شود تا به کمک آن ها گره را حذف کند. این تابع در دو تابع deleteLeftChild و deleteRightChild فراخوانی می‌شود که می‌توانید آن‌ها را در کد بررسی کنید. پارامتر های این تابع به این صورت است:

theNode: گره مورد نظر که قصد حذف کردن آن را از درخت داریم.

theParent: پدر گره مورد حذف که برای حذف درست به آن نیاز داریم.

```
    // Only leaf nodes and nodes with degree 1 can be deleted.
    // If a degree-1 node is deleted, it is replaced by its subtree.
    virtual void    deleteNode(IBTN* theNode, IBTN* theParent) {
        // Write your code here
    }
```

* توجه کنید که در یک درخت دودویی تنها حق حذف کردن گره های برگ را داریم و برنامه نباید اجازه حذف گرهی از وسط درخط را بدهد. پس با توجه به این نکته در صورتی که گره مورد نظر درجه یک یا بالاتر از آن را داشت (فرزند داشت) باید با پرتاب کردن exception ای از ادامه برنامه جلوگیری نمایید.

۵. حال باید عملگر های مربوط به iterator ها را پیاده‌سازی کنید. تمام iterator ها دارای دو عملگر زیر هستند که هر کدام براساس نوع خود به نحو ویژه‌ای عمل می‌کنند. پس باید با درنظر گرفتن نوع پیمایشگر عملگرش را به درستی پیاده کنید. (در حالت عادی برای هر کدام از این عملگر ها دو نوع post و pre وجود دارد که در این تمرین از post آن صرف نظر شده است)

```
        virtual void operator++(){
            // Write your code here
        }

        virtual void operator--(){
            // Write your code here
        }
```

** عملگر های مورد استفاده دیگر که در تمام iterator ها به صورت مشترک عمل می‌کردند در کلاس پدر یا همان iterator قرار گرفته است که توصیه می‌شود آن‌ها را مطالعه کنید تا با نوع کارکردشان آشنا گردید.

۶. در نهایت باید توابع مربوط به begin و end پیمایشگر ها را پیاده‌سازی نمایید. در این توابع لازم است با توجه به نوع پیمایشگر و با کمک گرفتن از dummy root ها گره درست را شناسایی کرده و آن را برگردانید. توجه کنید که در سی توابع begin به اولین عضو و توابع end به یک گره بعد از آخرین عضو اشاره می‌کند.
*** برای این قسمت نیز توصیه اکید می‌شود تا dummy root ها را از درس مطالعه فرمایید تا به مبحث مسلط گردید.

توجه: بهتر است جهت راحتی در پیاده‌سازی این تمرین، به ترتیب گفته شده شروع به تکمیل موارد کنید.
**** نکته مهم: در این تمرین برای صدا زدن ویژگی های mParents و mCurrentNode باید از this کمک بگیرید. به عنوان مثال برای استفاده از mCurrentNode به این صورت آن را فراخوانی نمایید:

```
this->mCurrentNode
```

نحوه نمره دهی به این سوال بدین ترتیب است:

۱- testInsertion: در این تست توابع مربوط به insert تست می‌شوند و ۱۲٪ از نمره را به خود اختصاص داده‌اند. (موارد ۲ و ۳)

۲- testDeletion: تست حذف کردن گره که شامل ۱۲٪ نمره است. (مورد ۴)

۳- تست های مربوط به پیمایشگر های زیر که هر کدام ۱۲٪ از نمره را دارند. (مورد ۵ و ۶)

testInOrderIteration, tesRevtInOrderIteration, testPreOrderIteration, testRevPostOrderIteration

۴- تست های مربوط به پیمایشگر های زیر که هر کدام ۱۴٪ از نمره را دارند. (مورد ۵ و ۶)

testRevPreOrderIteration, testPostOrderIteration

* دقت داشته باشید که مورد ۱ پیشنیاز بقیه موارد، به خصوص مورد ۶، است و بدون پیاده‌سازی آن نمی‌توانید بقیه موارد را به درستی و با روش صحیح آماده کنید. برای همین تست مخصوص به خودش را ندارد ولی در تست های دیگر تاثیر مستقیم دارد و بررسی می‌گردد.

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 به پوشه src بروید و پروژه را در Visual Studio باز کنید.
۳- کد های برنامه را تکمیل کنید.
٤- برای تست برنامه cmake را دانلود و نصب کنید. (در هنگام نصب مطمئن شوید cmake را به PATH ویندوز اضافه کرده باشید.)
برای دانلود می‌توانید از این لینک استفاده کنید:https://cmake.org/download/
٥- در ترمینال مطمئن شوید cmake و gcc نصب شده باشند. (می‌توانید از دو دستور cmake –version و gcc -v استفاده کنید.)
٦- افزونه CMake Tools را به vscode اضافه کنید.
۷- پوشه test را به تنهایی در vscode باز کنید.
۸- با Ctrl+Shift+P گزینه CMake: Select a Kit را انتخاب کرده و از آنجا کامپایلر gcc را انتخاب کنید.
۹- از سمت چپ، منو CMake را انتخاب کنید و از قسمت بالایی آن Build All Projects را انتخاب کنید. (همچنین می‌توانید با Ctrl+Shift+P و انتخاب گزینه CMake: Build نیز این کار را انجام دهید)
۱۰-حال با راست کلیک برروی تست کامپایل شده گزینه Run in Terminal را انتخاب کنید. (می‌توانید این مراحل را از این لینک نیز دنبال کنید: https://code.visualstudio.com/docs/cpp/cmake-linux)
۱۱-در صورت تمایل تست های بیشتری بنویسید تا از عملکرد برنامه خود اطمینان بیشتری حاصل نمایید.
۱۲- برنامه را اجرا کنید و پس از اطمینان از صحت عملکرد آن، با استفاده از کلید PrtScr از خروجی برنامه عکس بگیرید.
۱۳- عکس را با استفاده از mspaint در پوشه img ذخیره نمایید.

۱۴- 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 را حتما پاک کنید زیرا حجم زیادی می گیرند).

۱۵- محتویات کل پوشه را به صورت یک فایل zip در آورید.

۱۶-مطمئن شوید که وقتی فایل zip را باز می کنید پوشه های src, img و test و همچنین فایل info.txt را می بینید.

۱۷- نام این فایل zip را به «شماره تمرین-شماره دانشجویی» تغییر دهید.

۱۸-ابتدا این فایل را به سیستم «سپهر» ایمیل کنید تا از نحوه عملکر برنامه خود بر روی تست های تکمیلی آگاه شوید.

۱۹-اشکالاتی را که سیستم «سپهر» مشخص کرده است برطرف نمایید و مجددا تمرین را به سیستم «سپهر» تحویل دهید.

۲۰-مرحله قبل را آن قدر ادامه دهید که از صحت عملکرد برنامه خود اطمینان حاصل نمایید.

۲۱-نسخه نهایی فایل zip خود را تهیه نمایید.

۲۲- این فایل را از طریق سیستم vu تحویل دهید.

با آرزوی موفقیت