برنامه‌سازی پیشرفته **تمرین شماره‌ی شش**

مدرسان: رامتین خسروی، محمدامین صادقی طراحان: بهزاد اوسط، محمدرضا کیانی، سعید زنگنه

مهلت تحویل: پنج‌شنبه ۲۰ اردیبهشت‌ماه ۱۳۹۷، ساعت ۲۳:۵۵

هدف از این تمرین استفاده از مفاهیم وراثت و چندریختی در دو مسئله‌ی کوچک است .پس طراحی و استفاده‌ی صحیح از این عناصر در این پروژه از اهمیت ویژه‌ای برخوردار است.

درخت عبارت

درخت عبارت از تعدادی عملگر[1] و عملوند[2] تشکیل شده است. برگ‌های درخت عملوندهایی در مبنای ۱۰ و از نوع integer هستند و سایر گره‌های درونی درخت عملگرها خواهند بود. در این مسئله چهار عملگر جمع، ضرب،منفی و میانه را پیاده سازی خواهید نمود. از بین این چهار عملگر جمع و ضرب را دومتغیره، منفی را تک متغیره و میانه را چند متغیره فرض کنید و پس از ایجاد درخت عبارت، اطمینان حاصل کنید که این محدودیت‌های مطرح شده رعایت شده باشد. مثلا برای یک عملگر دو متغیره دقیقا دو فرزند قرار داده شده باشد. به سه مورد (یا بیشتر) از خطاهای احتمالی به دلخواه خود با استفاده از Exception Handling رسیدگی کنید و در صورت نیاز پیام خطای مناسب را به کاربر نمایش دهید.

add(x,y)	mult(x,y)	not(x)	med(x,y,z,...)
x+y	x*y	-x	میانه‌ی مجموعه‌ی ورودی

برای محاسبه‌ی حاصل یک درخت عبارت از برگ‌ها حرکت می‌کنیم و عملگرها را در هر سطح اعمال می‌کنیم تا به ریشه برسیم. مقدار مورد نظر برابر حاصل عملگر ریشه خواهد بود.

با توجه به فایل‌های tree.hpp و main_expression.cpp که در اختیار شما قرار گرفته‌است برنامه‌ای بنوسید که حاصل یک درخت عبارت را طبق چیزی که ذکر شد محاسبه کند. به نکات زیر درباره‌ی این فایل‌ها دقت فرمایید:

- متدهای add_operator_node و add_number_node مربوط به اضافه کردن گره به درخت می‌باشند که هر کدام سه آرگومان ورودی می‌گیرند. آرگومان اول اندیس گره و آرگومان دوم اندیس پدر آن خواهد. آرگومان سوم در هر یک به ترتیب نوع عملگر و مقدار عملوند می‌باشد.
- متد evaluate نیز مقدار نهایی درخت را محاسبه کرده و بازمی‌گرداند.
- این متدها را خودتان باید پیاده‌سازی نمایید.
- شما مجاز هستید کلاس‌های اضافه‌ی مورد نیاز خود را تعریف کنید و یا متدها و فیلدهایی به کلاس Tree اضافه کنید.

[1] Operator
[2] Operand

ECE Royale

در این مسئله قرار است که بازی ECE Royale را پیاده سازی کنید.

این بازی به صورت ۲ نفره اجرا می‌شود و هر بازیکن قبل از شروع بازی چهار کارت غیر تکراری انتخاب می‌کند و این تنها تعامل بازیکن با بازی خواهد بود.

با آغاز بازی کارت‌های انتخاب شده هر بازیکن به ترتیب و یکی پس از دیگری در مقابل هم قرار گرفته و به مبارزه می‌پردازند. پس از این مبارزه یکی از کارت‌ها حذف شده و در ادامه کارت بعدی از بازیکنی که کارتش حذف شده برای نوبت بعد وارد بازی می‌شود. این روند ادامه پیدا کرده و در هر نوبت یک کارت حذف می‌شود تا جایی که فقط یک کارت در بازی وجود داشته باشد که بازیکن برنده را مشخص می‌کند.

هر کارت دارای سه خصیصه‌ی قدرت[3] و هوشمندی[4] و چابکی[5] می‌باشد که مقداری بین ۰ تا ۱۰۰ دارند و در کل کارت‌ها به سه دسته تقسیم می‌شوند: Agility, Intelligence, Strength که هر کدام در توانمندی متناظر خود قوی‌ترند.

در بازی ۶ قهرمان وجود دارد که بازیکنان کارت‌های خود را از میان آن‌ها انتخاب می‌کنند. در جدول شماره‌ی ۱ نام و میزان هر خصیصه‌ی آن‌ها مشخص شده است.

در جدول شماره‌ی ۲ نحوه‌ی مبارزه و قوانین مشخص شده است. با توجه به اینکه در هر راند یک کارت حذف خواهد شد، در هر راندی که شرایط جدول برای حذف هیچ یک از کارت‌ها کافی نبود،کار را به سرنوشت واگذار کرده و یک کارت را به صورت تصادفی حذف کنید.

برای راحت‌تر شدن کار شما برای این مسئله فایل‌های game.hpp و main_royale.cpp در اختیارتان قرار داده شده‌اند که حاوی ساختار اصلی تابع main و کلاس Game می‌باشند. به نکات زیر حتما توجه کنید:

- یک کلاس Game تعریف شده است که دارای دو متد add_player_card و play می‌باشد که باید خودتان پیاده‌سازی کنید.
- متد add_player_card دو آرگومان ورودی می‌گیرد که اولی شماره‌ی بازیکن (۱ یا ۲) و دومین آرگومان نام کارت (از جدول ۱) خواهد بود.
- متد play نیز راندهای بازی را از ابتدا تا انتها اجرا می‌کند و **در انتها فقط شماره‌ی بازیکن برنده را چاپ می‌کند.**
- شما مجاز هستید هر کلاس دیگری برای برنامه‌ی خود تعریف کنید. طراحی خود را به درستی انجام دهید و برای هر کلاس خود دلیل مناسب داشته باشید.
- همچنین مجاز هستید بخش‌های جدیدی به کلاس Game اضافه کنید.

تاکید می‌شود که هدف از پروژه طراحی و استفاده‌ی صحیح از مفاهیم وراثت وچندریختی می‌باشد و استفاده از if یا switch case و ... برای تشخیص گره‌های درخت در مسئله‌ی اول یا نوع کارت‌ها در مسئله‌ی دوم قابل قبول نخواهد بود.

[3] strength
[4] intelligence
[5] agility

نام	نوع	strength	agility	intelligence
dragon	Strength	۷۰	۵۰	۳۰
giant	Strength	۸۵	۳۰	۳۵
archer	Agility	۳۵	۷۵	۴۰
goblin	Agility	۵۰	۹۰	۱۰
witch	Intelligence	۴۰	۳۰	۸۰
ghost	Intelligence	۴۰	۴۰	۷۰

جدول شماره‌ی ۱

نوع	توضیحات
A vs A	هرکدام که مجموع strength و intelligence بیشتری داشته باشد برنده خواهد بود و ۵ واحد intelligence و ۵ واحد strength دریافت می‌کند.
A vs I	هر کدام که strength بیشتری داشته باشد برنده خواهد بود و برنده ۱۰ واحد intelligence و ۱۰ واحد agility دریافت می‌کند.
A vs S	کارت از نوع Agility حذف خواهد شد و برنده ۱۰ واحد agility دریافت می‌کند.
S vs S	مجموع intelligence و agility برنده را مشخص خواهد کرد و برنده ۱۰ واحد intelligence دریافت می‌کند.
S vs I	هرکدام که agility بیشتری داشته باشد برنده خواهد بود و برنده ۱۰ واحد strength دریافت می‌کند.
I vs I	اگر تفاوت intelligenceها بیشتر یا مساوی ۲۰ باشد، کارت با intelligence بیشتر برنده خواهد بود. در غیر این صورت مجموع دو خصیصه‌ی agility و strength برنده را تعیین می‌کند و برنده ۵ واحد agility و ۵ واحد strength دریافت می‌کند.

جدول شماره‌ی ۲. هر یک از حروف S و A و I به ترتیب بیان‌گر انواع Strength ، Agility و Intelligence می‌باشند.

نحوه‌ی تحویل

تمامی فایل‌های مساله‌ی اول و دوم را به ترتیب در **دو پوشه با نام‌های p1 و p2** قرار داده و این پوشه‌ها را در آرشیوی با نام A6-SID.zip در صفحه‌ی CECM درس بارگذاری کنید؛ برای مثال، اگر شماره‌ی دانشجویی شما ۸۱۰۱۱۲۳۴۵ است، نام فایل شما باید A6-810112345.zip باشد.

- در صورتی که موارد گفته شده درباره‌ی نام‌گذاری فایل‌های خود را رعایت نکنید، نمره‌ی تست تمرین را از دست خواهید داد.
- برنامه‌ی شما باید در سیستم‌عامل لینوکس و با مترجم g++ قابل اجراباشد و فرمت فایل‌های شما همانند مشخصات گفته شده باشد، در غیر این صورت تمامی نمره‌ی تست تمرین را از دست خواهید داد.
- از صحت فرمت ورودی‌ها و خروجی‌های برنامه‌ی خود مطمئن شوید.
- هدف تمرین‌های درس یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.