**پروژه۲ کامپایلر**

هدف از پروژه‌های کامپایلر، پیاده‌سازی اجزاء مختلف یک کامپایلر واقعی برای نسخة محدود شده‌ای از زبان جاوا (که به آن زبان **میوا** -مخفف مینیمال جاوا- می‌گوییم) می‌باشد. سینتکس زبان میوا کاملا مشابه با زبان جاوا می‌باشد که توضیح آن به دقت در صورت سوال پروژه قبلی آورده شده است.

**پروژه دوم** در مورد پیاده سازی پارسر کامپایلر برای زبان میوا می‌باشد. برنامه شما یک فایل را از ورودی دریافت کرده و در صورت درست بودن فایل از نظر تحلیل‌گر لغوی (که در پروژه قبلی نوشته شده)، توکن‌های موجود در فایل را بازمی‌گرداند. سپس پارسر لیست توکن‌ها را دریافت نموده و **الگوریتم نزولی بازگشتی** را که در کلاس توضیح داده شد بر روی آن اجرا می‌کند. در این پروژه، نیازی به ساختن درخت پارس نیست و وظیفه اصلی پارسر تشخیص این نکته است که آیا برنامه خطای پارسری دارد و یا خیر. **حتما و حتما** ابتدا گرامر مستقل از زبان خود را برای زبان میوا نوشته و سپس رفع ابهام کنید و آنرا به عنوان بخش مهمی از این پروژه به همراه کد **آپلود کرده و تحویل دهید.**

**نکته مهم ۱:** اگر در برنامه خطائی از نظر پارسر وجود داشته باشد، شماره خطی که خطا در آن رخ داده باید در خروجی گزارش شود. نیازی به گزارش کلیه خطاهای پارسری در برنامه نیست و فقط کافیست اولین خطا گزارش شود. بنابراین در صورت برخوردن به یک خطای پارسری دیگر نیازی به ادامه اجرای کامپایل نیست.

**نکته مهم ۲**: برای سادگی رفع ابهام از گرامر، می‌توانید فرض کنید که بدنه if و بدنه else تنها با استفاده از { } قابل تعریف است، یعنی به صورت: if (condition) {//body} else {//else body} و یا if (condition) {//body}. مابقی قواعد سینتکس زبان میوا کاملا شبیه زبان جاوا می‌باشد.

**نکته مهم ۳:** کد خود را فقط و فقط در داخل فایلهای نمونه ارسال شده به گروه (توسط آقای زارع زاده) بنویسید و توابع مشخص شده را در داخل آنها پیاده سازی کنید. البته میتوانید بسته به نیاز خود فایلها، کلاسها و یا توابع دیگری نیز به فایلهای نمونه اضافه کنید. کلاس اصلی پروژه دوم کلاس Parser می‌باشد که دارای یک تابع parse برای پارس توکن‌های تشخیص داده شده توسط Lexer است. وظیفه اصلی شما پیاده سازی کامل تابع E() برای پارس عبارات از لیست توکن‌های شناسایی شده است و اینکار را صرفا باید با استفاده از الگوریتم پارس نزولی بازگشتی و توابع و قواعد تشریح شده در کلاس انجام دهید (مثلا همانطور که در کلاس گفته شد، تابع term برای شناسایی ترمینالها از ورودی مورد استفاده قرار می‌گرفت).

مثلا اگر گرامر زبان را به صورت قاعده تولید زیر در نظر بگیریم:

program🡪int main {E}

آنگاه تابع parse خود تابع program را فراخوانی می‌کند و تابع program به شکل زیر پیاده سازی می‌شود:

term(“int”, “keyword”) && term(“main”, “keyword”) && term("{", “separator”) && E() && token("}", “separator”);

البته کد فوق ساده سازی شده بدین معنی که در شکل دقیق آن باید گرامر زیر پیاده سازی شود:

program🡪int main() {E}