هدف: طراحی تحلیلگر لغوی (Lexical Analyzer) برای یک زبان برنامه نویسی شبه C

فاز اول پروژه شامل پیاده‌سازی واحد تحلیلگر لغوی است که در اینجا Scanner نامیده می‌شود. وظیفه اصلی واحد Scanner استخراج تمام Token های موجود در یک زبان برنامه‌نویسی شبه C می‌باشد. این برنامه با دریافت یک فایل ورودی به زبان C باید تمام توکن‌ها را در خروجی نمایش دهد. این برنامه می‌تواند شامل ساختارها و متدهای زیر باشد.

- **ساختاری برای معرفی انواع توکن‌های مورد نیاز که با توجه به مطالب مطرح شده پیشنهاد می‌گردد از نوع Enum تعریف گردد و شامل انواع زیر می‌باشد.**
  - توکن id
  - توکن number
  - توکن Special Token (ST) شامل ... =< < => > { ( ) } ; ++ +
  - توکن String
  - توکن Comment
  - توکن keyword
- **ساختاری برای تعریف یک توکن که دارای دو فیلد می‌باشد.**
  - فیلد type معرفی کننده نوع توکن از نوع Enum قسمت قبل.
  - فیلد value برای نگهداری مقادیر مورد نیاز هر توکن از نوع String.
- **جدولی از تمام keyword های زبان مانند:**
  - if
  - while
  - do
  - for
  - main
  - return
  - int
  - float
  - double
  - char
  - else
- **تابع جستجوی findKeyword(String str) که با دریافت یک string به دنبال آن در جدول keyword ها می‌گردد. درصورت یافت شدن true و در غیر اینصورت false برمی‌گرداند.**

- تابع **nextToken()** که در واقع پیاده‌سازی اصلی از برنامه **Scanner** بوده و شامل پیاده‌سازی **DFA** زبان سطح اول برنامه‌نویسی می‌باشد. این تابع شامل یک دستور **switch case** (چنانچه در کلاس درس مطرح گردیده است) می‌باشد.
- تابعی که با فراخوانی تابع **nextToken()** از ابتدا تا انتهای فایل را به صورت توکن در خروجی نمایش می‌دهد. به عنوان نمونه قطعه کد زیر را به صورت نمایش داده شده در خروجی نشان می‌دهد.

ورودی:

```
main ( )
{
        int a, b;
         if (a == b )
        {
                a = a*2 ;
       }
}
```

خروجی:

<KW, ‘main’> <ST, ‘(’> <ST, ‘)’> <ST, ‘{’> <KW, ‘int’> <id, ‘a’> <ST, ‘,’> <id, ‘b’> <ST, ‘;’>

<KW, ‘if’> <ST, ‘(’> <id, ‘a’> <ST, ‘==’> <id, ‘b’> <ST, ‘)’> <ST, ‘{’> <id, ‘a’> <ST, ‘=’> <id, ‘a’> <ST, ‘*’> <number, ‘2’> <ST, ‘;’> <ST, ‘}’> <ST, ‘}’>

مهلت تحویل پروژه چهارشنبه دوم آبان