

1. Do Problem 7.19(c).

The sample code in the text has an abstract class named `AArg` for abstract argument with one method named `generateCode()`. Only this single method is needed for the example language to produce both the listing and the object code.

In Pep/9 assembly language, the translator will need two methods, `generateListing()`, which returns a string value of the object to be output for the listing, and `getIntValue()`, which returns the integer value to be used for the code generator. In this parsing phase of your project, you only need to implement `generateListing()`.

Here is a sample run with some valid input.

Parser input

```
br 0x17, i
.BLOCK 0x3
.block 1
deci 0x03,d
LDWA 0x0003,d
adda 0x9, i

deco 0xaB, s
stop
.end
```

Parser output

```
Program Listing
BR      0x0017
.BLOCK  0x0003
.BLOCK  1
DECI    0x0003,d
LDWA    0x0003,d
ADDA    0x0009,i

DECO    0x00AB,s
STOP
.END
```

Here is a sample run with some invalid input.

Parser input

```
brr 0x7, i
.BLOCK 0xG
deci 0x03, e
```

Parser output

```
ERROR: Illegal mnemonic  
ERROR: Illegal hex constant  
ERROR: Illegal addressing mode  
ERROR: Missing .END sentinel
```

Your program will be tested more extensively than the short examples above. Be sure to test it thoroughly.

Restriction: For all phases of this project you may not use any Java library functions that parse strings, for example `Integer.parseInt()`. That would defeat the purpose of the program, which is to parse the source.

Name your Java package `prob0719`. Note the lowercase `p`. Create a separate source file for each Java class. The first line of your source files must be `package prob0719;`. Name your IntelliJ project `Prob0719` and the class that has the main program as `Prob0719Main` in a file named `Prob0719Main.java`.

Export the source files in a JAR file named `Prob0719.jar`. Hand in the `.jar` file.