(1) O objetivo do trabalho é implementar um compilador para a Linguagem Lovelace, usando tradução dirigida por sintaxe, na ferramenta Javacc. Para realizar esta tarefa, o aluno deve adicionar ações semânticas às regras sintáticas já implementadas em Javacc para que seja gerada uma árvore sintática do programa Lovelace sendo compilado. Em seguida, o programa deve percorrer a árvore sintática gerando código (**na linguagem C**), semanticamente equivalente ao código Lovelace original. As dicas de como implementar esse compilador estão na **vídeo aula: 12 - Tradução Dirigida Por Sintaxe usando o Javacc**.

As classes a serem usadas para a árvore sintática estão disponíveis para download no e-aula (arquivo **ArvoreSintatica.zip**). A explicação sobre as classes está no arquivo **Árvore Sintática.pdf**. **É obrigatório o uso dessas classes na construção da árvore sintática**.

O main do javacc, deve ficar mais ou menos assim:

```
// importar as classes da árvore sintática:

import ast.*;

public class Lovelace {

  public static void main(String args[]) throws Exception{
   // abrir o arquivo passado por linha
   // de comando contento o código em Lovelace:

  FileInputStream fs = new FileInputStream(new File(args[0]));

   // Instanciar o parser da linguagem Lovelace passando
   // como argumento o arquivo contendo o código
   // Lovelace a ser processado:

    Lovelace parser = new Lovelace(fs);

   // Chamar a primeira regra do parser que irá
   // analisar o código e devolver a árvore sintática

   Prog arvore = parser.Lovelace();

  // passar a árvore para o gerador de código
  // que deve gerar um arquivo com com o código
  // alvo na linguagem C:

   geraCodigo(arvore, args[0])

  }

 public static void geraCodigo(Prog prog, String arquivo){??????}


}
```

Além do compilador escrito em Javacc (**Lovelace.jj**) o aluno deve desenvolver **dois exemplos de programas** que usem uma grande parte das construções sintáticas disponíveis na linguagem (incluindo funções).