**Compiler Construction**

Develop a translator for the **Ruby++** language. Translator will generate three address code.

1. **Required Submissions:**
   1. Compiler Code, which actually contains **Lex, Parser & Translator integrated in single compiler.**
2. **Translation Requirements:**Translator should be able to generate three address code for following statements in program:
   1. **print**
   2. **gets**
   3. **if else**
   4. **while loop**
   5. **return**
   6. **function declaration**
   7. **expression and assignment**
   8. **arrays (one-dimensional only)**
3. **Input format for Compiler:**

You should follow below mentioned sequence for successful parsing.

* 1. **Lex** takes input of **Ruby++** code from **file** named **input.txt.**
  2. **Lex** generates output in file named **lex**\_**output.txt.**
  3. **Parser** takes generated output of **Lex** as an input.
  4. **Parser** generates parse tree in file named **parse\_tree.txt** up to valid input code and print parsing error if there is any.
  5. **Translator** generates three address code in file named **thress\_address\_code.txt**.
  6. Steps (1 – 5) will be performed automatically, User only runs your compiler and all required steps are performed automatically if user has placed **input.txt** correctly, which actually contains **Ruby++** code.

1. **Input Output Sample:**

**Sample Input:**

**int i, j**

**i = 0**

**j = 0**

**while i < 10**

**{**

**j = j + i \* i**

**i = i + 1**

**}**

**Output of Input:**

**L1:t1 := 0 ; initialize i**

**L2:t2 := 0 ; initialize j**

**L3: if t1 < 10 gotoL5; conditional jump**

**L4:gotoL10 ; square of i**

**L5: t3 := t1 \* t1 ; square of i**

**L6:t4 := t2 + t3 ; add j + square of i**

**L7: t2 := t4; assign result of expression to j**

**L8:t1 := t1 + 1 ; increase i**

**L9:gotoL3 ; repeat loop**

**L10:**

**Important Notes:**

* **Please go through mentioned sample input and output. Your program should take input code of Ruby++ and generates all required output files.**
* **Every Assignment will add up and makes one more step towards complete running Compiler. i.e.  
  1st submission: Program contains (Lex)**

**2nd submission: Program contains (Lex + Parser)**

**3rd submission: Program contains (Lex + Parser + Translator)**

**etc.**

* **Your all assignments are actually sequential, so every next assignment will be dependent on output of previous assignment.So, you are unable to do current assignment without all previous assignments.**
* **JPlag will be used to detect plagiarism and will results minimum F in Compiler Construction course.**