**471/672 In-class Group Exercises**

**Getting Started with Generating TAC Code from Decaf**

**Why?**

**Learning Objectives:**

1. Familiarize yourself with the requirements for PP5
2. Familiarize yourself with the starter files (CodeGenerator, Tac, and Mips classes)
3. Begin devising strategies for:

* Assigning locations to variables
* Generating instructions for expressions and control structures

**Resources:**

1. PP5 assignment sheet
2. PP5 starter files
3. Handout with t1.decaf, t1.tac, t1.asm

**Tasks:**

1. In your group, assign team roles:
   1. **Manager** - ensuring that everyone is fulfilling their roles, that assigned tasks are being accomplished and that all members of the team are participating in the activity? This is the only person that may ask questions or have contact with the facilitator.
   2. **Reader** - responsible for reading out loud so everyone on the team stays together for the activity
   3. **Recorder** - responsible for writing down important ideas/concepts and ensuring that the rest of the team is completing the answers on their activity sheet
   4. **Presenter** - responsible for reporting to the facilitator and the rest of the class when asked
2. As a group, answer the following key questions. The recorder should type the answers in a file that can be shared with the whole group and used by the Presenter to explain to the class.
   1. What is the input and output of your code generator? What is your responsibility in generating the output of the code generator?
   2. What are the main methods that you will be writing to implement your code generator?
   3. List several simplifications of the Decaf language that have been made for code generation? That is, what do you not have to handle during code generation?
   4. What are the command lines to run the Decaf compiler to see the tac generated for a given Decaf program? Mips output?
   5. What size are each of the variables to be allocated by your code generator?
   6. What would the Location object contain for a global variable named myGlobal that is the third global variable?
   7. What would the Location object contain for the first local variable, which is named myFirstLocal?
   8. What would the Location object contain for the second parameter, which is named mySecondParm?
   9. Where does the frame pointer point to in the stack?
   10. What is the first parameter in every method call?
3. Using the handout with t1.decaf, t1.tac, and t1.asm, draw lines to segment which instructions in t1.tac and t1.asm correspond to different instructions in t1.decaf.
4. Explain the 24 in BeginFunc. How is it computed?
5. Using the mapping from above, develop a sequence of calls to predefined methods for you in codegen and tac that would need to be called by the code generator to implement each Decaf instruction in t1.decaf, and note any other decisions that need to be made during that code generation.
6. Determine when you will figure out and fill in the Location object for each variable – global, local, and parameters.
7. Discuss and outline a plan for how to proceed in starting your code generator.