Let us assume that **ONLY** a small number of user registers (**3**) is available on a given RISC machine.
This assumption defies the principle that "a large number of registers" is one of the basic key elements in the design of RISC processors; in principle, this exercise would apply even if there were more registers available.
When compiling a HLL, optimized register usage is the responsibility of the compiler.
A program written in a high-level language has, of course, no explicit references to registers.
Rather, program values/variables are referred to symbolically.
The objective of the compiler is to keep the operands for as many computations as possible in registers rather than main memory, and to minimize the number of load-and-store operations associated.

The following program segment in (a certain) Assembly language, is the result of a 'compilation' from a HLL, using typical translation templates for a certain CISC Instruction Set.
Now, the target Processor is a 'RISC' machine, therefore further translation is required to convert it into a Reduced Instruction Set.
The 'RISC' processor is strictly a 'LOAD/STORE' machine, which means that ONLY the 'LOAD/STORE' instructions are used to access memory into/from registers and arithmetic/logic operations ONLY use registers.
The 'Templates' used for the translation are:

```
ADD A, B, C ➔ LR1   A
              LR2   B
              ADD R1, R2, R3 (adds R1 and R2 and puts result in R3)
              STR3 C
```

BRanches are not modified.

```
START  ADD  A, B, C      (add operands A and B, stores result in operand C)
       ADD  D, E, F
       ADD  C, F, G
       BRZ  ISZERO
       BRN  ISNEG
       ADD  F, G, C
       BR END
ISZERO ADD  F, J, C
       BR END
ISNEG  ADD  G, J, C
END    STOP
```

---

1) Translate the given CISC program segment into RISC using the provided templates (LAB4-CODE-TEMPLATE).
If done correctly, you should have 29 instructions (6 adds @4 each plus 4 br@1 each plus 1 stop).

2) NOW, Optimize the resulting RISC code by re-arranging the register usage and eliminating unnecessary Load/Store Instructions. Use the concepts presented in section 13.3 to determine the 'time sequence of active use of registers' (fig 13.4 a) for their re-use/re-arrangement.
The register re-use/re-arrangement will in turn suggest/inform the re-arrangement/removal of redundant 'Load/Store' instructions. It is suggested that the number of instructions should be reduced to 20(or less).

3) Now VERIFY that the semantics of the original program HAS BEEN PRESERVED, which should be done by comparing the results of the execution of the original CISC program(provided) and the OPTIMIZED RISC program results, using different sets of values for the input variables so as to exercise all three conditional branches.

When completed, upload the completed (LAB4-CODE-TEMPLATE).