Example Solutions - Lecture 20

1. The GNU Compiler Collection (gcc) compiler suite offers many levels of optimizations. Investigate the various levels that are available for a given language (e.g. C++). Hint: look at gcc flags controlling optimizations (Fischer et. al. exercise 1 on page 668 in GE)

   Resources about the optimizations available in GCC and LLVM:

   a. GCC 2.95.3: http://gcc.gnu.org/onlinedocs/gcc-2.95.3/gcc_2.html#SEC10
   b. GCC 9.3.0: https://gcc.gnu.org/onlinedocs/gcc-9.3.0/gcc/Optimize-Options.html
   c. Clang 11: https://clang.llvm.org/docs/UsersManual.html
   d. LLVM's Passes: https://llvm.org/docs/Passes.html

   How different combinations of compilers, flags, and platforms change the instructions produced for a specific program is easy to investigate using the Compiler Explorer at: https://godbolt.org/

Group Exercises - Lecture 20

1. Discuss the outcome of the individual exercises
   Did you all agree on the answer?

2. The GCC compiler suite can compile Java programs from either source or bytecode form. What are the advantages and disadvantages of compiling from source as compared to compiling from bytecode form? (Fischer et. al. 3, on page 668 in GE)
   While source code is more difficult to parse than bytecode, it often contains more information about the program that the compiler can use for optimizations. For example, at the bytecode level different constructs in a programming language are often represented by similar sequences of bytecodes, making it harder to distinguish between them if an optimization only applies to one of the constructs. In addition, some concepts might not even exist at the bytecode level as the source-to-bytecode compiler removes them, e.g., through the use of type erasure.