Introduction to Language Theory and Compilation
Solutions
Session 8: Code generation

Solutions

Ex. 1.
```
@y = constant i32 3

define i32 @exercise1(){
   entry:
      %x = call i32 @readInt()
      %0 = load i32* @y
      %1 = add i32 %x, 3
      %2 = sub i32 9, %0
      %3 = mul i32 %1, %2
      ret i32 %3
}
```

Ex. 2.
```
define void @exercise2(){
   entry:
      ; item 1
       %a = alloca i32
       %b = alloca i32
      ; item 2
      %0 = call i32 @readInt()
      %1 = call i32 @readInt()
      store i32 %0, i32* %a
      store i32 %0, i32* %b
      ; item 3
      %2 = load i32* %a
      %3 = add i32 5, %2
      store i32 %3, i32* %a
      ; item 4
      %4 = load i32* %b
      %5 = sdiv i32 %4, 2
      store i32 %5, i32* %b
      ; item 5
      %6 = load i32* %a
      %7 = load i32* %b
      %8 = icmp sgt i32 %6, %7
      br i1 %8, label %output\_a, label %output\_b
   output\_a:
      %9 = load i32* %a
      call void @println(i32 %9)
      ret void
   output\_b:
      %10 = load i32* %b
      call void @println(i32 %10)
      ret void
}
```

Ex. 3.
```
define i32 @readInt() {
   entry:                                    ; create variables
      %res   = alloca i32
      %digit = alloca i32
      store i32 0, i32* %res
      br label %read
   read:                                              ; read a digit
      %0          = call i32 @getchar()
      %1 = sub i32 %0, 48
```

```
      store i32 %1, i32* %digit
      %2 = icmp ne i32 %0, 10
      br i1 %2, label %save, label %exit
   save:                                             ; res<-res*10+digit
      %3 = load i32* %res
      %4 = load i32* %digit
      %5 = mul i32 %3, 10
      %6 = add i32 %5, %4
      store i32 %6, i32* %res
      br label %read
   exit:                                             ; return res
      %7 = load i32* %res
      ret i32 %7
}
```

Ex. 4.

```
; External declaration of the getchar function
declare i32 @getchar()
; External declaration of the putchar function
declare i32 @putchar(i32);

; External declaration for ramdomization
declare i32 @rand()
declare i32 @time(i32*)
declare void @srand(i32)

define i32 @getNumber(){
   entry:
      %0 = call i32 @rand()
      %1 = srem i32 %0, 100
      ret i32 %1
}

define i32 @main() {
    ;initialization of randomizer
    %1 = call i32 @time(i32* null)
    call void @srand(i32 %1)
    %2 = call i32 @getNumber()
    %guess = alloca i32
    store i32 %2, i32* %guess
    %i = alloca i32
    store i32 0, i32* %i
    br label %for_loop
   for_loop:
        %3 = load i32* %i
        %4 = icmp slt i32 %3, 5
        br i1 %4, label %inner_for, label %after_for
   inner_for:
      %try = alloca i32
        %5 = call i32 @readInt()
        store i32 %5, i32* %try
      %6 = load i32* %try
      %7 = load i32* %guess
      %8 = icmp sgt i32 %6, %7
      br i1 %8, label %first_cond, label %test_next
   test_next:
        %9 = load i32* %try
      %10 = load i32* %guess
      %11 = icmp slt i32 %9, %10
      br i1 %11, label %second_cond, label %else
   first_cond:
        ;greater
        call i32 @putchar(i32 45);-
      call i32 @putchar(i32 10);\n
      br label %end_for
    second_cond:
      ;lower
      call i32 @putchar(i32 43);+
```

```
        call i32 @putchar(i32 10);\n
        br label %end_for
    else:
        ;success
        call i32 @putchar(i32 79);O
        call i32 @putchar(i32 75);K
        call i32 @putchar(i32 10);\n
        ret i32 0
        br label %end_for ; never reached
    end_for:
          %20 = load i32* %i
          %21 = add i32 %20, 1
          store i32 %21, i32* %i
          br label %for_loop
    after_for:
          ;failure
        call i32 @putchar(i32 75);K
        call i32 @putchar(i32 79);O
        call i32 @putchar(i32 10);\n
        ret i32 0
}
```