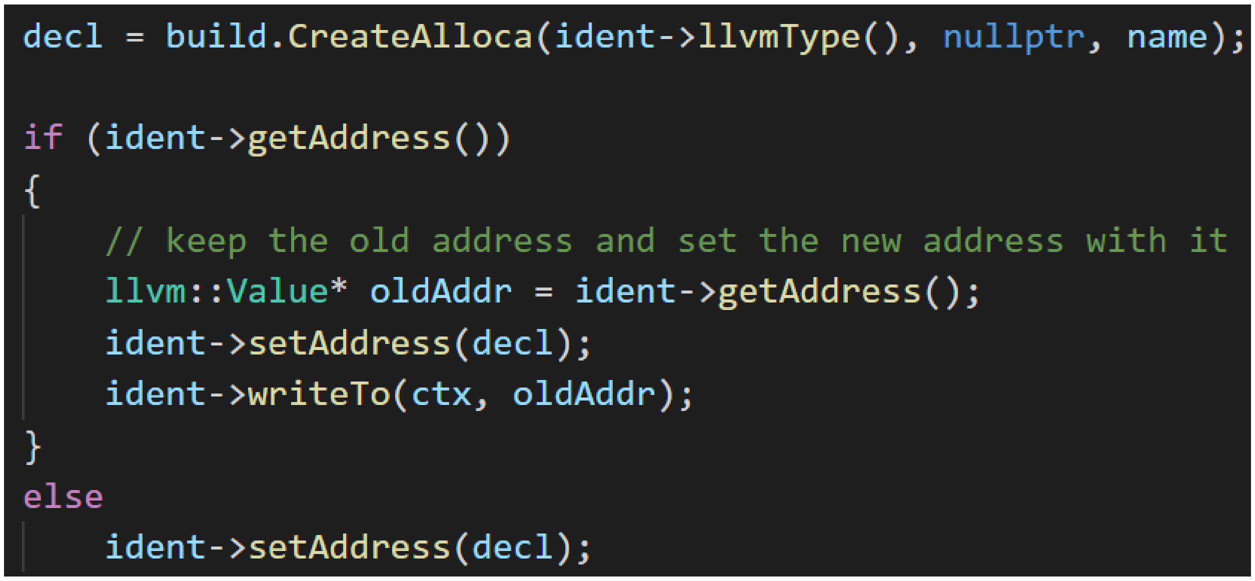
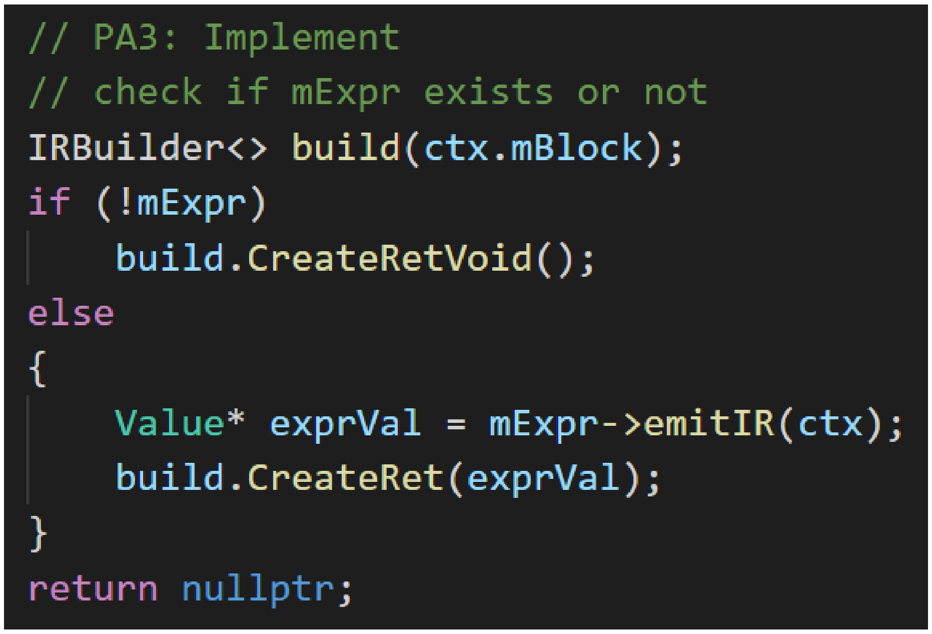
The following are my code snippets and comments for each emit-related function I implemented. I just list a few functions that make me struggle while debugging. Thanks again for TA's help in debugging part.

- **Symbols.cpp/emitIR()**
  We only focus on non-array cases in this function. When handling the special case, function parameter, I was confused by the meaning of "value" contained in the address. After clarification, I figure out that we need to set the new address as the old address in the new allocated space.



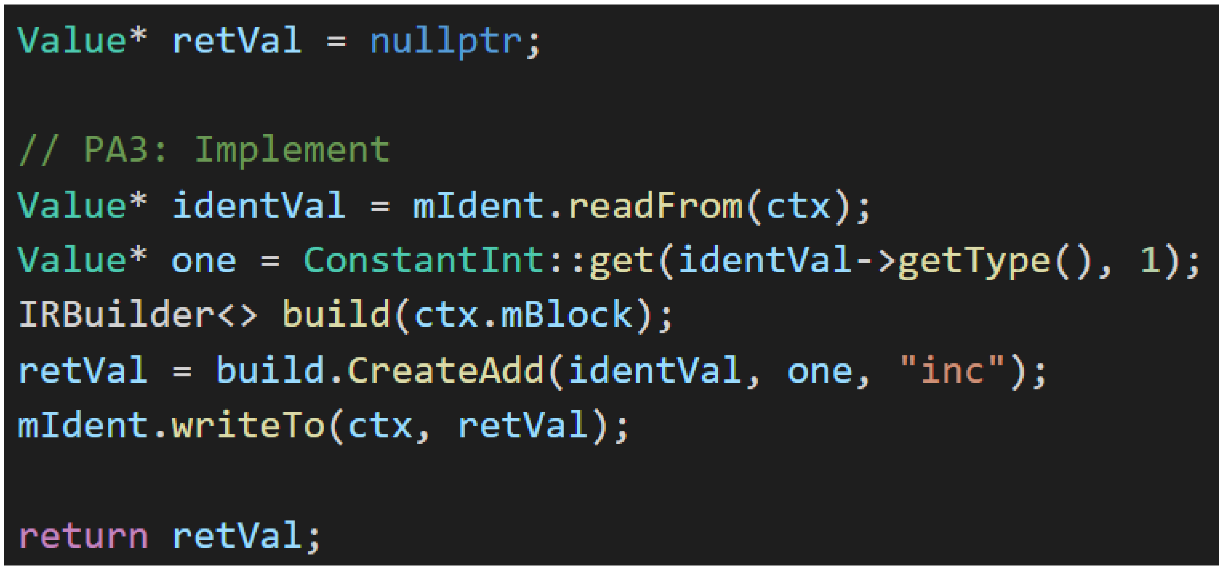
```
decl = build.CreateAlloca(ident->llvmType(), nullptr, name);

if (ident->getAddress())
{
    // keep the old address and set the new address with it
    llvm::Value* oldAddr = ident->getAddress();
    ident->setAddress(decl);
    ident->writeTo(ctx, oldAddr);
}
else
    ident->setAddress(decl);
```

- **ASTEmit.cpp/(ASTReturnStmt)**
  The function deals with return statements. I did not notice that it may dereferences a null pointer when there is no expression after return. Anyway, the bug is found with a lot of efforts.



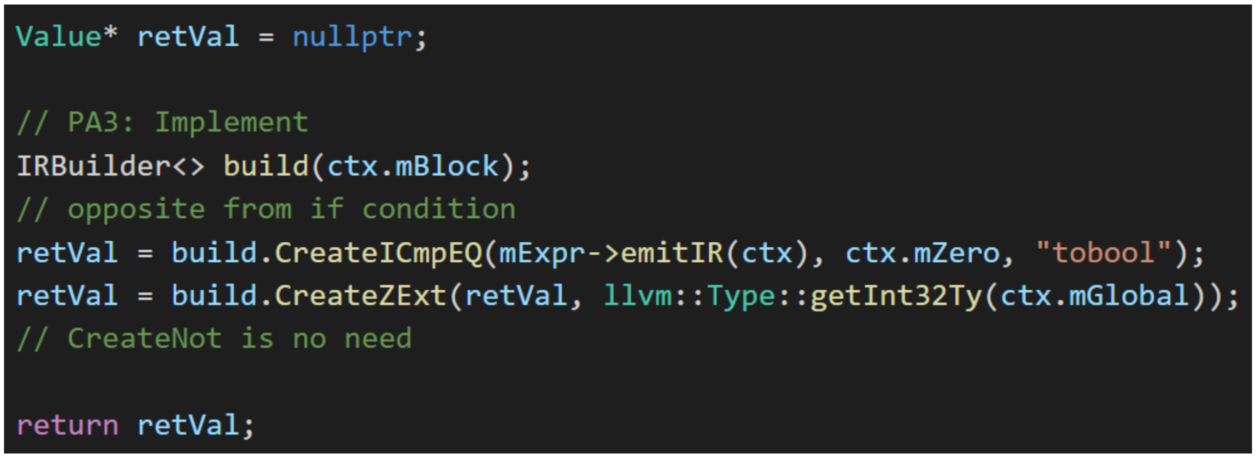
```
// PA3: Implement
// check if mExpr exists or not
IRBuilder<> build(ctx.mBlock);
if (!mExpr)
    build.CreateRetVoid();
else
{
    Value* exprVal = mExpr->emitIR(ctx);
    build.CreateRet(exprVal);
}
return nullptr;
```

- **ASTEmit.cpp/(ASTIncExpr/ASTDecExpr)**
  The function reads the value of the identifier and adds/subs 1 to it and store the updated value. I did not notice that the value returned from build needs to be returned.



```
Value* retVal = nullptr;

// PA3: Implement
Value* identVal = mIdent.readFrom(ctx);
Value* one = ConstantInt::get(identVal->getType(), 1);
IRBuilder<> build(ctx.mBlock);
retVal = build.CreateAdd(identVal, one, "inc");
mIdent.writeTo(ctx, retVal);

return retVal;
```

- **ASTEmit.cpp/(ASTNotExpr)**
  This function evaluates the expression and check if the value is zero or not. I misadded a instruction which turned the value from positive to negative.



```
Value* retVal = nullptr;

// PA3: Implement
IRBuilder<> build(ctx.mBlock);
// opposite from if condition
retVal = build.CreateICmpEQ(mExpr->emitIR(ctx), ctx.mZero, "tobool");
retVal = build.CreateZExt(retVal, llvm::Type::getInt32Ty(ctx.mGlobal));
// CreateNot is no need

return retVal;
```