compiler_hw4 R08922195

Build code

In src folder, run

```
make
```

Run Test case

In run_and_main, run

```
./run.sh test.c
```

Assignment statements and Arithmetic expressions

- Float assignment and arithmetic

```
./run.sh float_expr.c
```

- Int assignment and arithmetic

```
./run.sh int_expr.c
```

Control statements: while, if-then-else

- Float control

```
./run.sh float_control.c
```

- Int control

```
./run.sh int_control.c
```

Parameterless procedure calls

- Self define function

```
./run.sh float_function_call.c
./run.sh int_function_call.c
```

```
./run.sh
```

Read and Write I/O calls (only support int and float)

- Read function

```
./run.sh read.c
```

Part 2: Constant folding

```
./run.sh constant_folding.c
```

In semanticAnalysis.c, when we visit a exprNode, we check whether its right and left children can be constant evaluated. If its childrens are both can be constant evaluated, then we directly compute from its children and store to the parent node. Then, we label the parent node as a constant node. After we visit all the expression tree, we already compute all the possible constant folding. The following code demostrate the concept.

```
void fold(AST_NODE* exprNode){
  AST_NODE *left = exprNode->left;
  AST_NODE *right = exprNode->right;
  fold(left);
  fold(right);
  if(isConstant(left) && isConstant(right) ){
    exprNode->val = compute(left, right);
    exprNode->isConstant = 1;
  }
}
```