interp class, method interp(Stm s)

```
static void interp(Stm s) {
   Table t = interpStm (s, new Table ("", 0, null));
 }
static Table interpStm (Stm s, Table t) {
   if (s instanceof PrintStm)
     return interpPrintedExps (((PrintStm) s).exps, t);
   else if (s instanceof AssignStm) { /* remember as.exp might have Stms! */
     AssignStm as = (AssignStm) s;
     IntAndTable it  = interpExp (as.exp, t);
     return update (it.t, as.id, it.i);
   }
   else if (s instanceof CompoundStm) {
     CompoundStm cs = (CompoundStm) s;
     return interpStm (cs.stm2, interpStm (cs.stm1, t));
   }
   else {
     System.out.println ("error in interpStm");
     return t;
   }
}
```

interp class, method interp(Stm s)

```
static Table interpPrintedExps (ExpList e, Table t) {
  if (e instanceof PairExpList) {
    PairExpList pe = (PairExpList) e;
    IntAndTable it = interpExp (pe.head, t);
    System.out.print (it.i);
    System.out.print (" ");
    return interpPrintedExps (pe.tail, it.t);
  }
  else if (e instanceof LastExpList) {
    IntAndTable it = interpExp (((LastExpList) e).head, t);
    System.out.println (it.i);
    return it.t;
  }
  else {
    System.out.println ("error in interpPrintedExps");
    return t;
  }
}
```

interp class, method interp(Stm s)

```
static IntAndTable interpExp (Exp e, Table t) {
  if (e instanceof IdExp)
    return new IntAndTable (lookup (t, ((IdExp) e).id), t);
  else if (e instanceof NumExp)
    return new IntAndTable (((NumExp) e).num, t);
  else if (e instanceof OpExp) {
    OpExp oe = (OpExp) e;
    IntAndTable it1 = interpExp (oe.left, t); IntAndTable it2 = interpExp (oe.right, it1.t);
    if (oe.oper == OpExp.Plus)  return new IntAndTable (it1.i+it2.i, it2.t);
    else if (oe.oper == OpExp.Minus) return new IntAndTable (it1.i-it2.i, it2.t);
    else if (oe.oper == OpExp.Times) return new IntAndTable (it1.i*it2.i, it2.t);
    else if (oe.oper == OpExp.Div) return new IntAndTable (it1.i/it2.i, it2.t);
    else throw new Error ("interpExp: oper not recognised");
  }
  else if (e instanceof EseqExp)
    return interpExp (((EseqExp) e).exp, interpStm (((EseqExp) e).stm, t));
  else
    throw new Error ("interExp: exp not recognised");
}
```

method int maxargs(Stm s)

```
static int maxargs(Stm s) {
  if (s instanceof PrintStm)
    return Math.max (maxargs (((PrintStm) s).exps),
                              length (((PrintStm) s).exps));
  else if (s instanceof AssignStm)
    return maxargs (((AssignStm) s).exp);
  else if (s instanceof CompoundStm)
    return Math.max (maxargs (((CompoundStm) s).stm1),
                              maxargs (((CompoundStm) s).stm2));
  else {
    System.out.println ("maxargs(Stm): unrecognised Stm");
    return 0;
  }
}
```

method int maxargs(Stm s)

```
static int maxargs (ExpList e) {
  if (e instanceof PairExpList)
    return Math.max (maxargs (((PairExpList) e).head), maxargs (((PairExpList) e).tail));
  else if (e instanceof LastExpList)
    return maxargs (((LastExpList) e).head);
  else {
    System.out.println ("maxargs(ExpList): unrecognised ExpList");
    return 0;
  }
}
```

method int maxargs(Stm s)

```
static int maxargs (Exp e) {
  if (e instanceof OpExp)
    return Math.max (maxargs (((OpExp) e).left), maxargs (((OpExp) e).right));
  else if (e instanceof EseqExp)
    return Math.max (maxargs (((EseqExp) e).stm), maxargs (((EseqExp) e).exp));
  else /* it's an IdExp or a NumExp */
    return 0;
}
```