レポート

システムソフトウェア特論 課題#3

担当教員 久野靖

提出日 2020 年 2 月 17 日

電気通信大学
情報理工学研究科
情報・ネットワーク工学専攻

学籍番号 1931010

伊勢大平

1 課題

選んだ演習問題は 3-3 である．a,b,c,d,e,f の全てを解いた．
第 2 節 (課題の再掲) で，課題を要約して説明する．

2 課題の再掲

ノード群 (Lit，Var，Add，Mul，Sub，Div，Mod，Eq，Ne，Gt，Ge，Lt，Le，And，Or，Not，Assign，Seq，Read，Print，While，If1) に，次のような機能を持つノードを追加し，プログラムを抽象構文木で組み立てて動作を確認する．

a. do-while ループを実現するノード DoWhile(ループ本体のノード，条件式のノード)．

b. else 部のある if 文を実現するノード If2(条件のノード，then 部のノード，else 部のノード)．

c. 回数を指定してループ本体をその回数だけ繰り返すノード Times(回数の式のノード，ループ本体のノード)．

d. 上記と同様だが，周回ごとに指定した変数が 0,1,2,$\cdots$ と変化していくノード TimesFor(回数の式のノード，変数のノード，ループ本体のノード)．

e. 式を指定してその式の値に応じてどれかの選択肢を実行するノード Switch(式のノード，new Node[]$\{B_1, B_2, \ldots, B_n\}$)．式の値が 1 なら B_1，2 なら B_2，等が実行される．どれでもない場合はどれも実行されない．

f. 自分で作ってみたいと思うノード $\cdots$ 二つの式の値を比較して，大小関係に応じてどれかの選択肢を実行するノード GorEorL(比較の式 1 のノード，比較の式 2 のノード，実行の式 1 のノード，実行の式 2 のノード，実行の式 3 のノード)．比較の式 1 が比較の式 2 と比べて小さければ実行の式 1，等しければ実行の式 2，大きければ実行の式 3 が実行される．

3　方針

prog のノードの作成は授業で習った方法と同じである．作成した prog ノードの記述によってどのような動作が起こるかは，第 4.2 節 (プログラムの説明) で述べる．

追加する各ノードは，次のような方針で作成した．

DoWhile ループ本体のノードを一度実行し，その後に while 文で条件を満たす限り実行するようにする．

If2 Java の if - else の構文を利用して作る．

Times for 文を用いる．for 文のループ回数を k とし，k に回数の式のノードの値を代入する．

TimesFor 第二引数で受け取った変数の変数名を使って，for 文の中で，vars のその変数名のところにループ変数の値を代入する．ループ変数は 0,1,2,$\cdots$ と変化する．

Switch 第一引数の式のノードを child に add した後，第二引数以降のノードを順次 child に add する．child の 0 番目の値（＝ 式のノードの値）を変数 i に代入し，child の i 番目のノードの値を返す．

GorEorL 比較の式 1 の値と比較の式 2 の値を比較し，比較結果に応じて実行の式の値を v に格納し，v を返す．

4 成果物

4.1 プログラム

プログラムはリスト 1 のように書いた.

リスト 1: ソースコード (Sam32.java)

```
import java.util.*;

public class Sam32 {
    static Map<String,Integer> vars = new TreeMap<String,Integer>();

    public static void main(String[] args) {
        Node prog = new Seq(
                            // Read k;
                            new Read(new Var("k")),
                            // Switch
                            new Switch(
                                      // case number of Switch
                                      new Var("k"),
                                      // k=1 (case 1 of Switch)
                                      new Seq(
                                             new Read(new Var("x")),
                                             new Assign(new Var("i"),new Lit(10)),
                                             // Do While
                                             new DoWhile(
                                                       // body of loop
                                                       new Seq(
                                                              new Assign(new Var("i"),
                                                                  new Add(new Var("i"),
                                                                  new Lit(2))),
                                                              new Print(new Var("i"))),
                                                       // cond of loop
                                                       new Lt(new Var("i"),new Var("x"))
                                                       )
                                             ),
                                      // k=2 (case 2 of Switch)
                                      new Seq(
                                             new Read(new Var("x")),
                                             // If2
                                             new If2(
                                                    // cond
                                                    new Ge(new Var("x"), new Lit(30)),
                                                    // then
                                                    new Print(new Lit(30)),
                                                    // else
                                                    // If2
                                                    new If2(
                                                           // cond
                                                           new Ge(new Var("x"), new Lit
                                                               (20)),
                                                           // then
                                                           new Print(new Lit(20)),
                                                           // else
                                                           // If2
                                                           new If2(
                                                                  // cond
                                                                  new Ge(new Var("x"),
                                                                      new Lit(10)),
                                                                  // then
                                                                  new Print(new Lit(10)),
                                                                  // else
                                                                  new Print(new Lit(0))
```

```
                                                                )
                                                            )
                                                        )
                                                ),
                                        // k=3 (case 3 of Switch)
                                        new Seq(
                                                new Read(new Var("x")),
                                                new Assign(new Var("i"), new Lit(0)),
                                                // Times
                                                new Times(
                                                        // number of loop
                                                        new Mod(new Var("x"), new Lit(5)),
                                                        // body of loop
                                                        new Assign(new Var("i"),new Add(new
                                                             Var("i"), new Lit(1)))
                                                        ),
                                                new Print(new Var("i"))
                                                ),
                                        // k=4 (case 4 of Switch)
                                        new Seq(
                                                new Read(new Var("x")),
                                                new Assign(new Var("i"), new Lit(1)),
                                                // TimesFor
                                                new TimesFor(
                                                        // number of loop
                                                        new Var("x"),
                                                        // var of TimesFor
                                                        new Var("y"),
                                                        // body of loop
                                                        new Seq(
                                                                new Assign(new Var("i"),
                                                                    new Mul(new Var("i"),
                                                                     new Add(new Var("y")
                                                                    , new Lit(1)))),
                                                                new Print(new Var("y")),
                                                                new Print(new Var("i"))
                                                                )
                                                        )
                                                ),
                                        // k=5 (case 5 of Switch)
                                        new Seq(
                                                new Read(new Var("x")),
                                                new Read(new Var("y")),
                                                new GorEorL(
                                                        // var1
                                                        new Var("x"),
                                                        // var2
                                                        new Var("y"),
                                                        // var1 < var2
                                                        new Print(new Sub(new Var("y"),
                                                            new Var("x"))),
                                                        // var1 == var2
                                                        new Print(new Lit(0)),
                                                        // var1 > var2
                                                        new Print(new Sub(new Var("x"),
                                                            new Var("y")))
                                                        )
                                                )
                                        )
                                );
        System.out.println(prog);
        //System.out.println(prog.eval());
        prog.eval();
    }
    abstract static class Node {
```

```
112         List<Node> child = new ArrayList<Node>();
113         public void add(Node n) { child.add(n); }
114         public abstract int eval();
115     }
116     static class Lit extends Node {
117         int val;
118         public Lit(int v) { val = v; }
119         public int eval() { return val; }
120         public String toString() { return ""+val; }
121     }
122     static class Var extends Node {
123         String name;
124         public Var(String n) { name = n; }
125         public int eval() { return vars.get(name); }
126         public String toString() { return name; }
127     }
128     abstract static class BinOp extends Node {
129         String op;
130         public BinOp(String o, Node n1, Node n2) { op = o; add(n1); add(n2); }
131         public String toString() { return "("+child.get(0)+op+child.get(1)+")"; }
132     }
133     static class Add extends BinOp {
134         public Add(Node n1, Node n2) { super("+", n1, n2); }
135         public int eval() { return child.get(0).eval() + child.get(1).eval(); }
136     }
137     static class Mul extends BinOp {
138         public Mul(Node n1, Node n2) { super("*", n1, n2); }
139         public int eval() { return child.get(0).eval() * child.get(1).eval(); }
140     }
141     static class Sub extends BinOp {
142         public Sub(Node n1, Node n2) { super("-", n1, n2); }
143         public int eval() { return child.get(0).eval() - child.get(1).eval(); }
144     }
145     static class Div extends BinOp {
146         public Div(Node n1, Node n2) { super("/", n1, n2); }
147         public int eval() { return child.get(0).eval() / child.get(1).eval(); }
148     }
149     static class Mod extends BinOp {
150         public Mod(Node n1, Node n2) { super("%", n1, n2); }
151         public int eval() { return child.get(0).eval() % child.get(1).eval(); }
152     }
153     static class Eq extends BinOp {
154         public Eq(Node n1, Node n2) { super("==", n1, n2); }
155         public int eval() { return child.get(0).eval()==child.get(1).eval()?1:0; }
156     }
157     static class Ne extends BinOp {
158         public Ne(Node n1, Node n2) { super("!=", n1, n2); }
159         public int eval() { return child.get(0).eval()!=child.get(1).eval()?1:0; }
160     }
161     static class Gt extends BinOp {
162         public Gt(Node n1, Node n2) { super(">", n1, n2); }
163         public int eval() { return child.get(0).eval()>child.get(1).eval()?1:0; }
164     }
165     static class Ge extends BinOp {
166         public Ge(Node n1, Node n2) { super(">=", n1, n2); }
167         public int eval() { return child.get(0).eval()>=child.get(1).eval()?1:0; }
168     }
169     static class Lt extends BinOp {
170         public Lt(Node n1, Node n2) { super("<", n1, n2); }
171         public int eval() { return child.get(0).eval()<child.get(1).eval()?1:0; }
172     }
173     static class Le extends BinOp {
174         public Le(Node n1, Node n2) { super("<=", n1, n2); }
175         public int eval() { return child.get(0).eval()<=child.get(1).eval()?1:0; }
176     }
```

```
177     static class And extends BinOp {
178         public And(Node n1, Node n2) { super("&&", n1, n2); }
179         public int eval() {
180             int v = child.get(0).eval();
181             if(v == 0) { return 0; } else { return child.get(1).eval(); }
182         }
183     }
184     static class Or extends BinOp {
185         public Or(Node n1, Node n2) { super("||", n1, n2); }
186         public int eval() {
187             int v = child.get(0).eval();
188             if(v != 0) { return v; } else { return child.get(1).eval(); }
189         }
190     }
191     abstract static class UniOp extends Node {
192         String op;
193         public UniOp(String o, Node n1) { op = o; add(n1); }
194         public String toString() { return "("+op+child.get(0)+")"; }
195     }
196     static class Not extends UniOp {
197         public Not(Node n1) { super("!", n1); }
198         public int eval() { return child.get(0).eval()==0?1:0; }
199     }
200     static class Assign extends Node {
201         Var v1; Node n1;
202         public Assign(Var v, Node n) { v1 = v; n1 = n; }
203         public int eval() {
204             int v = n1.eval(); vars.put(v1.toString(), v); return v;
205         }
206         public String toString() { return v1+"="+n1; }
207     }
208     static class Seq extends Node {
209         public Seq(Node... a) { for(Node n:a) { child.add(n); } }
210         public int eval() {
211             int v = 0;
212             for(Node n:child) { v = n.eval(); }
213             return v;
214         }
215         public String toString() {
216             String s = "{\n";
217             for(Node n:child) { s += " " + n.toString() + ";\n"; }
218             return s + "}";
219         }
220     }
221     static class Read extends Node {
222         Var v1;
223         public Read(Var v) { v1 = v; }
224         public int eval() {
225             System.out.print(v1+"? ");
226             Scanner sc = new Scanner(System.in);
227             String str = sc.nextLine();
228             int i = Integer.parseInt(str);
229             vars.put(v1.toString(), i); return i;
230         }
231         public String toString() { return "read "+v1; }
232     }
233     static class Print extends Node {
234         public Print(Node n1) { child.add(n1); }
235         public int eval() {
236             int v = child.get(0).eval(); System.out.println(v); return v;
237         }
238         public String toString() { return "print "+child.get(0); }
239     }
240     static class While extends Node {
241         public While(Node n1, Node n2) { child.add(n1); child.add(n2); }
```

```
242         public int eval() {
243             int v = 0;
244             while(child.get(0).eval() != 0) { v = child.get(1).eval(); }
245             return v;
246         }
247         public String toString() {
248             return "while("+child.get(0)+")"+child.get(1);
249         }
250     }
251     static class If1 extends Node {
252         public If1(Node n1, Node n2) { child.add(n1); child.add(n2); }
253         public int eval() {
254             int v = 0;
255             if(child.get(0).eval() != 0) { v = child.get(1).eval(); }
256             return v;
257         }
258         public String toString() { return "if("+child.get(0)+")"+child.get(1); }
259     }
260     static class DoWhile extends Node {
261         public DoWhile(Node n1, Node n2) { child.add(n1); child.add(n2); }
262         public int eval(){
263             int v = 0;
264             v=child.get(0).eval();
265             while(child.get(1).eval() != 0) { v = child.get(0).eval(); }
266             return v;
267         }
268         public String toString(){
269             return "do "+child.get(0)+" while "+child.get(1);
270         }
271     }
272     static class If2 extends Node {
273         public If2(Node n1, Node n2, Node n3) { child.add(n1); child.add(n2); child.add
                (n3); }
274         public int eval() {
275             int v = 0;
276             if(child.get(0).eval() != 0) { v = child.get(1).eval(); }
277             else { v = child.get(2).eval(); }
278             return v;
279         }
280         public String toString() { return "if ("+child.get(0)+") "+child.get(1)+" else
                "+child.get(2); }
281     }
282     static class Times extends Node {
283         public Times(Node n1, Node n2) { child.add(n1); child.add(n2); }
284         public int eval(){
285             int v = 0;
286             int k = child.get(0).eval();
287             for(int i=0;i<k;i++){ v=child.get(1).eval(); }
288             return v;
289         }
290         public String toString(){
291             return "times("+child.get(0)+") "+child.get(1);
292         }
293     }
294     static class TimesFor extends Node {
295         Var v1;
296         public TimesFor(Node n1, Var n2, Node n3) { child.add(n1); v1=n2; child.add(n3)
                ; }
297         public int eval(){
298             int v = 0;
299             int k = child.get(0).eval();
300             for(int i=0;i<k;i++){
301                 vars.put(v1.toString(), i);
302                 v=child.get(1).eval();
303             }
```

```
304             return v;
305         }
306         public String toString(){
307             return "timesfor("+child.get(0)+","+v1.toString()+") "+child.get(1);
308         }
309     }
310     static class Switch extends Node {
311         public Switch(Node n1, Node... a) { child.add(n1); for(Node n:a) { child.add(n)
                ; } }
312         public int eval(){
313             int v = 0;
314             int i = child.get(0).eval();
315             if(1<=i&&i<child.size()) { v = child.get(i).eval(); }
316             return v;
317         }
318         public String toString(){
319             String s = "switch("+child.get(0)+"){\n";
320             for(int i=1;i<child.size();i++){ s += i+" : "+child.get(i)+";\n"; }
321             return s+"}";
322         }
323     }
324     static class GorEorL extends Node {
325         public GorEorL(Node n1, Node n2, Node n3, Node n4, Node n5){
326             child.add(n1); child.add(n2); child.add(n3); child.add(n4); child.add(n5);
327         }
328         public int eval(){
329             int v=0;
330             if(child.get(0).eval()<child.get(1).eval()) v = child.get(2).eval();
331             else if(child.get(0).eval()==child.get(1).eval()) v = child.get(3).eval();
332             else if(child.get(0).eval()>child.get(1).eval()) v = child.get(4).eval();
333             return v;
334         }
335         public String toString(){
336             return "GorEorL (" + child.get(0).toString() + "," + child.get(1).toString()
                    + ") {\n" +
337                 " <: " + child.get(2).toString() + "\n" +
338                 " ==: " + child.get(3).toString() + "\n" +
339                 " >: " + child.get(4).toString() + "\n}";
340         }
341     }
342 }
```

4.2 プログラムの説明

Seq の先頭で変数 k の値を Read し，k の値を用いて Switch が実行される．

k=1 のとき DoWhile が実行される．変数 x の値を Read し，変数 i に 10 を代入する．DoWhile で i が x より小さい限り，「i の値を 2 増やして i の値を表示する」を繰り返す．DoWhile なので，「i の値を 2 増やして i の値を表示する」は必ず一回は実行される．

k=2 のとき If2 が実行される．変数 x の値を Read する．If2 の else 部のノードに If2 を使うことで，if - else 構文を実現できるので，その方法で次のように実行される．

1. x の値が 30 以上だったら 30 を表示する．
2. 30 以上でなく，20 以上だったら 20 を表示する．

3. 20 以上でもなく，10 以上だったら 10 を表示する．

4. 10 以上でもなかったら 0 を表示する．

k=3 のとき Times が実行される．変数 x の値を Read するし，変数 i に 0 を代入する．Times のループを用いて，x を 5 で割った余りの値の分だけ「i に 1 を足す」を繰り返す．i の値を表示する．結局，x を 5 で割った余りの値が表示される．

k=4 のとき TimesFor が実行される．変数 x の値を Read し，変数 i に 1 を代入する．TimesFor の変数（周回ごとに 0,1,2,$\cdots$ と変化する変数）を y とする．TimesFor のループを用いて，x の値の分だけ,「$i \times (y+1)$ の結果を i に代入し，y の値を表示し，i の値を表示する」を繰り返す．つまり，TimesFor の n 回目のループで $n-1$ と $n!$ が表示される．例えば変数 x の値が 5 なら 5! が最後に表示される．

k=5 のとき GorEorL が実行される．変数 x と変数 y の値を Read する．x と y の値を比較し，その結果により次のように実行される．

1. $x < y$ のとき，$y - x$ の値を表示する．
2. $x = y$ のとき，0 を表示する．
3. $x > y$ のとき，$x - y$ の値を表示する．

以上により，$|x - y|$ の値が表示される．

それ以外のとき どれも実行されない．

4.3 実行例

プログラムの実行結果はリスト 2,. . .,14 のようになった．

リスト 2 とリスト 3 では，最初の入力が 1 なので k=1 となり，DoWhile が実行される．

リスト 2 では，1 を入力したあとに 5 を入力しており，DoWhile の条件を満たさないので一度だけループ本体が実行され，i が 12 となって出力される．

リスト 3 では，1 を入力したあとに 17 を入力しており，DoWhile のループ本体が何度か実行され，i が 18 となって出力される．

以上より，DoWhile が正しく動作していると分かる．

リスト 2: 実行結果 (入力は 1 と 5)

```
1  {
2   read k;
3   switch(k){
4  1 : {
5   read x;
6   i=10;
7   do {
8   i=(i+2);
9   print i;
10 } while (i<x);
11 };
12 2 : {
```

```
13  read x;
14  if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
        10 else print 0;
15 };
16 3 : {
17  read x;
18  i=0;
19  times((x%5)) i=(i+1);
20  print i;
21 };
22 4 : {
23  read x;
24  i=1;
25  timesfor(x,y) {
26  i=(i*(y+1));
27  print y;
28  print i;
29 };
30 };
31 5 : {
32  read x;
33  read y;
34  GorEorL (x,y) {
35  <:  print (y-x)
36  ==: print 0
37  >:  print (x-y)
38 };
39 };
40 };
41 }
42 k? 1
43 x? 5
44 12
```

リスト 3: 実行結果 (入力は 1 と 17)

```
 1 {
 2  read k;
 3  switch(k){
 4 1 : {
 5  read x;
 6  i=10;
 7  do {
 8  i=(i+2);
 9  print i;
10 } while (i<x);
11 };
12 2 : {
13  read x;
14  if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
        10 else print 0;
15 };
16 3 : {
17  read x;
18  i=0;
19  times((x%5)) i=(i+1);
20  print i;
21 };
```

```
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
 ==: print 0
 >:  print (x-y)
};
};
};
}
k? 1
x? 17
12
14
16
18
```

リスト 4 とリスト 5 とリスト 6 とリスト 7 では，最初の入力が 2 なので k=2 となり，If2 が実行される．

リスト 4 では，2 を入力したあとに 100 を入力しており，一つ目の if 文の条件を満たし，30 が出力される．

リスト 5 では，2 を入力したあとに 21 を入力しており，二つ目の if 文の条件を満たし，20 が出力される．

リスト 6 では，2 を入力したあとに 10 を入力しており，三つ目の if 文の条件を満たし，10 が出力される．

リスト 7 では，2 を入力したあとに 9 を入力しており，一，二，三つ目の if 文の条件を満たさず，最後の else 文により 0 が出力される．

以上より，If2 が正しく動作していると分かる．また，if - else if - else if $\cdots$ と繋げても正しく動作していると分かる．

リスト 4: 実行結果 (入力は 2 と 100)

```
{
 read k;
 switch(k){
1 : {
 read x;
 i=10;
 do {
 i=(i+2);
 print i;
} while (i<x);
};
2 : {
```

```
13  read x;
14  if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
        10 else print 0;
15 };
16 3 : {
17  read x;
18  i=0;
19  times((x%5)) i=(i+1);
20  print i;
21 };
22 4 : {
23  read x;
24  i=1;
25  timesfor(x,y) {
26  i=(i*(y+1));
27  print y;
28  print i;
29 };
30 };
31 5 : {
32  read x;
33  read y;
34  GorEorL (x,y) {
35  <:  print (y-x)
36  ==: print 0
37  >:  print (x-y)
38 };
39 };
40 };
41 }
42 k? 2
43 x? 100
44 30
```

リスト 5: 実行結果 (入力は 2 と 21)

```
 1 {
 2  read k;
 3  switch(k){
 4 1 : {
 5  read x;
 6  i=10;
 7  do {
 8  i=(i+2);
 9  print i;
10 } while (i<x);
11 };
12 2 : {
13  read x;
14  if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
        10 else print 0;
15 };
16 3 : {
17  read x;
18  i=0;
19  times((x%5)) i=(i+1);
20  print i;
21 };
```

```
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
 ==: print 0
 >:  print (x-y)
};
};
};
}
k? 2
x? 21
20
```

リスト 6: 実行結果 (入力は 2 と 10)

```
{
 read k;
 switch(k){
1 : {
 read x;
 i=10;
 do {
 i=(i+2);
 print i;
} while (i<x);
};
2 : {
 read x;
 if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
      10 else print 0;
};
3 : {
 read x;
 i=0;
 times((x%5)) i=(i+1);
 print i;
};
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
```

```
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
 ==: print 0
 >:  print (x-y)
};
};
};
}
k? 2
x? 10
10
```

リスト 7: 実行結果 (入力は 2 と 9)

```
{
 read k;
 switch(k){
1 : {
 read x;
 i=10;
 do {
 i=(i+2);
 print i;
} while (i<x);
};
2 : {
 read x;
 if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
      10 else print 0;
};
3 : {
 read x;
 i=0;
 times((x%5)) i=(i+1);
 print i;
};
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
 ==: print 0
 >:  print (x-y)
};
};
};
}
```

```
42 k? 2
43 x? 9
44 0
```

リスト 8 とリスト 9 では，最初の入力が 3 なので k=3 となり，Times が実行される．

リスト 8 では，3 を入力したあとに 13 を入力しており，13 を 5 で割った余りの 3 が出力される．

リスト 9 では，3 を入力したあとに 15 を入力しており，15 を 5 で割った余りの 0 が出力される．

以上より，Times が正しく動作していると分かる．

リスト 8: 実行結果 (入力は 3 と 13)

```
 1 {
 2  read k;
 3  switch(k){
 4 1 : {
 5  read x;
 6  i=10;
 7  do {
 8  i=(i+2);
 9  print i;
10 } while (i<x);
11 };
12 2 : {
13  read x;
14  if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
       10 else print 0;
15 };
16 3 : {
17  read x;
18  i=0;
19  times((x%5)) i=(i+1);
20  print i;
21 };
22 4 : {
23  read x;
24  i=1;
25  timesfor(x,y) {
26  i=(i*(y+1));
27  print y;
28  print i;
29 };
30 };
31 5 : {
32  read x;
33  read y;
34  GorEorL (x,y) {
35  <:  print (y-x)
36  ==: print 0
37  >:  print (x-y)
38 };
39 };
40 };
41 }
42 k? 3
43 x? 13
```

```
44 3
```

リスト 9: 実行結果 (入力は 3 と 15)

```
 1 {
 2  read k;
 3  switch(k){
 4 1 : {
 5  read x;
 6  i=10;
 7  do {
 8  i=(i+2);
 9  print i;
10 } while (i<x);
11 };
12 2 : {
13  read x;
14  if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
       10 else print 0;
15 };
16 3 : {
17  read x;
18  i=0;
19  times((x%5)) i=(i+1);
20  print i;
21 };
22 4 : {
23  read x;
24  i=1;
25  timesfor(x,y) {
26  i=(i*(y+1));
27  print y;
28  print i;
29 };
30 };
31 5 : {
32  read x;
33  read y;
34  GorEorL (x,y) {
35  <:  print (y-x)
36  ==: print 0
37  >:  print (x-y)
38 };
39 };
40 };
41 }
42 k? 3
43 x? 15
44 0
```

リスト 10 では，最初の入力が 4 なので k=4 となり，TimesFor が実行される．4 を入力したあとに 5 を入力しており，最終的に 5! = 120 が出力される．なお，

- y が 0 のときは，y の値 0 と 1! として 1 が出力され，
- y が 1 のときは，y の値 1 と 2! の計算結果 2 が出力され，

- y が 2 のときは，y の値 2 と 3! の計算結果 6 が出力され，
- y が 3 のときは，y の値 3 と 4! の計算結果 24 が出力され，
- y が 4 のときは，y の値 4 と 5! の計算結果 120 が出力される．

TimesFor が正しく動作していると分かる．

リスト 10: 実行結果 (入力は 4 と 5)

```
{
 read k;
 switch(k){
1 : {
 read x;
 i=10;
 do {
 i=(i+2);
 print i;
} while (i<x);
};
2 : {
 read x;
 if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
      10 else print 0;
};
3 : {
 read x;
 i=0;
 times((x%5)) i=(i+1);
 print i;
};
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
 ==: print 0
 >:  print (x-y)
};
};
};
}
k? 4
x? 5
0
1
1
2
```

```
2
6
3
24
4
120
```

リスト 11 とリスト 12 とリスト 13 では，最初の入力が 5 なので k=5 となり，GorEorL が実行される．

リスト 11 では，5 を入力したあとに 13 と 13 を入力しており，$|13-13|=0$ が出力される．

リスト 12 では，5 を入力したあとに 13 と 17 を入力しており，$|13-17|=4$ が出力される．

リスト 13 では，5 を入力したあとに 17 と 13 を入力しており，$|17-13|=4$ が出力される．

以上より，GorEorL が正しく動作していると分かる．

リスト 11: 実行結果 (入力は 5 と 13 と 13)

```
{
 read k;
 switch(k){
1 : {
 read x;
 i=10;
 do {
 i=(i+2);
 print i;
} while (i<x);
};
2 : {
 read x;
 if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
      10 else print 0;
};
3 : {
 read x;
 i=0;
 times((x%5)) i=(i+1);
 print i;
};
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
```

```
 ==: print 0
 >:  print (x-y)
};
};
};
}
k? 5
x? 13
y? 13
0
```

リスト 12: 実行結果 (入力は 5 と 13 と 17)

```
{
 read k;
 switch(k){
1 : {
 read x;
 i=10;
 do {
 i=(i+2);
 print i;
} while (i<x);
};
2 : {
 read x;
 if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
      10 else print 0;
};
3 : {
 read x;
 i=0;
 times((x%5)) i=(i+1);
 print i;
};
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
 ==: print 0
 >:  print (x-y)
};
};
};
}
k? 5
x? 13
y? 17
```

```
45 4
```

リスト 13: 実行結果 (入力は 5 と 17 と 13)

```
 1 {
 2  read k;
 3  switch(k){
 4 1 : {
 5  read x;
 6  i=10;
 7  do {
 8  i=(i+2);
 9  print i;
10 } while (i<x);
11 };
12 2 : {
13  read x;
14  if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
       10 else print 0;
15 };
16 3 : {
17  read x;
18  i=0;
19  times((x%5)) i=(i+1);
20  print i;
21 };
22 4 : {
23  read x;
24  i=1;
25  timesfor(x,y) {
26  i=(i*(y+1));
27  print y;
28  print i;
29 };
30 };
31 5 : {
32  read x;
33  read y;
34  GorEorL (x,y) {
35  <:  print (y-x)
36  ==: print 0
37  >:  print (x-y)
38 };
39 };
40 };
41 }
42 k? 5
43 x? 17
44 y? 13
45 4
```

リスト 14 では，最初の入力が 6 なので k=6 となり，Switch のどのケースにも当てはまらず，なにもしないまま終了する．

また，リスト 2 からリスト 13 までで，Switch によって正しく分岐が行われていると分かる．

以上より，Switch が正しく動作していると分かる．

リスト 14: 実行結果 (入力は 6)

```
{
 read k;
 switch(k){
1 : {
 read x;
 i=10;
 do {
 i=(i+2);
 print i;
} while (i<x);
};
2 : {
 read x;
 if ((x>=30)) print 30 else if ((x>=20)) print 20 else if ((x>=10)) print
      10 else print 0;
};
3 : {
 read x;
 i=0;
 times((x%5)) i=(i+1);
 print i;
};
4 : {
 read x;
 i=1;
 timesfor(x,y) {
 i=(i*(y+1));
 print y;
 print i;
};
};
5 : {
 read x;
 read y;
 GorEorL (x,y) {
 <:  print (y-x)
 ==: print 0
 >:  print (x-y)
};
};
};
}
k? 6
```

5　考察

リスト 1 のソースコードの 107 行目の「System.out.println(prog);」がどういう意味なのか，はじめは分からなかったが,「System.out.println(prog.toString());」としても同じ動作をしたので，toSrting() の呼び出しを省略したものだと考えた．なぜ toString() を省略しても良いのか考えてみたが,「toString() はインスタンスを打ち出すなどのときに文字列に変換する際に呼び出される」ということが資料に書いてあり，インスタンス prog を println で打ち出そうとしていて，文字列に変換する必要があるため，この状況に当てはまっているのだと考察した．そのため，省略していても toString() は呼び出されると考えられる．

リスト 1 のソースコードの 109 行目の「prog.eval();」であるが，これはもともと 108 行目のコメント部分のように「System.out.println(prog.eval());」となっていた．eval() 自体で出力まで行うように抽象構文木が作られているので，最終的な eval() の返り値を出力しないようにと考えて，このように変更した．

toString() が実行されたとき，できればインデントをちゃんとしたいと考えた．そこで，Seq の toString() で空白を出力するようにしたり，GorEorL の toString() で空白を出力するようにしたりした．ある程度は見やすくなったが，全体をうまくインデントすることはできなかった．入れ子の構造のときにインデントがうまくできなかった．これを解決する方法が分からず，toString() の定義だけで解決するのは難しいのではないかと考えた．

6　アンケートの解答

Q1 コンパイラの構成についてどれくらい知っていましたか．また，どの部分にとくに興味がありますか．

A1 以前，実験で，C 言語を使って，yacc と lex を用いて，ソースコードを入力としてアセンブリを出力するプログラムを作りました．その時の記憶として，式の計算などが構文木によって行われることは覚えていました．また，構文木に伴って文脈自由文法の知識が要求されることも知っていました．

とくに興味があるのは，C 言語のポインタ機能や，Java のオブジェクト指向の機能（オブジェクトや継承）などです．これらの機能がどのような仕組みで，どのようにして作られているのか興味がわきました．

ある高級言語を設計したとき，それのコンパイラを既に完成している別の高級言語を用いて作成することが可能だということは理解できますが，一番最初のコンパイラをどうやって作ったのか疑問に思いました．ＯＳの作成も同様で，既に作られたＯＳ上でＯＳを作成することが可能なのは理解できますが，何もＯＳがない状態からＯＳを作る場合はどうすればよいのか疑問に思いました．

Q2 抽象構文木でプログラムを組み立ててみてどのように思いましたか．

A2 式や文の列が一つの木で表されることが理解できました．また，自分で作ってみたいと思うノードを追加できるというところに面白さを感じました．また，木を組み立てるときはどんどん深くなっていくので，リスト 1 のソースコードの 7 行目から 106 行目までの木の構成を書くところが大変だと思いました．

Q3 リフレクション（課題をやってみて気づいたこと），感想，要望など．

A3 ノードの追加は，例が多かったので真似するようにして簡単にできました．講義を受けて例を実行するだけでなく，プログラムを変更したり付け加えたりすることで理解が深まることを実感できました．課題 a,b,c,d,e,f で作ったノードを全て試す必要があったため，実行結果が多くなってしまいました．