```
/*
 *  Program for Pass I of Two-Pass Assembler
 */

package SPOS;

class symtab{
      int index;
      String name;
      int addr;


      symtab(int i, String s, int a){
            index = i;
            name = s;
            addr = a;
      }
}

class littab{
      int index;
      String name;
      int addr;

      littab(int i, String s, int a){
            index = i;
            name = s;
            addr = a;

      }

      void setaddr(int a) {
            addr = a;
      }
}

class pooltab{
      int p_index;
      int l_index;

      pooltab(int i, int a){
            p_index = i;
            l_index = a;
      }
}


public class pass1 {

      public static void main(String args[]) {

            String input[][] = {
                        {null, "START", "100", null},
                        {null, "MOVER", "AREG", "A"},
            {"AGAIN", "ADD", "AREG", "='2"},
            {null, "ADD", "AREG", "B"},

            {"AGAIN", "ADD", "AREG", "='3"},
            {null, "LTORG", null, null},

            {"AGAIN2", "ADD", "AREG", "BREG"},

            {"AGAIN2", "ADD", "AREG", "CREG"},

            {"AGAIN", "ADD", "AREG", "='2"},
            {null, "DC", "B", "3"},
            {"LOOP", "DS", "A", "1"},
            {null, "END", null, null}
};

symtab s[] = new symtab[20];
littab l[] = new littab[20];
pooltab p[] = new pooltab[20];

int loc=0, i=0;
String m, op1, op2;

int sn=0, ln=0, lnc=0, pn=0;

loc = Integer.parseInt(input[0][2]);

m = input[1][1];
i = 1;

while (!m.equals("END")) {
      if (check(m) == 1) {
            if (input[i][0] == null) {
                  op1 = input[i][2];
                  op2 = input[i][3];
                  if (comp(op2, s, sn) == 1) {
                        s[sn] = new symtab(sn, op2,  0);
                        sn++;
                  }
                  else if (comp(op2, s, sn) == 2) {
                        l[ln] = new littab (ln,op2, 0);
                        ln++;
                  }
                  loc++;
                  i++;
            }
            else {
                  op1 = input[i][0];
                  s[sn] = new symtab(sn, op1,loc);
                  sn++;

                  op1 = input[i][2];
                  op2 = input[i][3];
```

```
                        if (comp(op2, s, sn) == 1) {
                              s[sn] = new symtab(sn, op2, 0);
                              sn++;
                        }
                        else if (comp(op2, s, sn) == 2) {
                              l[ln] = new littab(ln, op2, 0);
                              ln++;
                        }
                        loc++;
                        i++;
                  }
            }
            else if (check(m) == 2) {
                  if(input[i][0] == null) {
                        int temp;
                        op1 = input[i][2];
                        op2 = input[i][3];
                        temp = comps(op1, s, sn);
                        if (temp != 99) {
                              s[temp] = new symtab(temp, op1, loc);

                        }
                        loc = loc + Integer.parseInt(op2);
                        i++;
                  }
                  else {
                        int temp;
                        op1 = input[i][0];
                        s[sn] = new symtab(sn, op1, loc);
                        sn++;

                        op1 = input[i][2];
                        op2 = input[i][3];
                        temp = comps(op1, s, sn);
                        if (temp != 99) {
                              s[temp] = new symtab(temp, op1, loc);
                        }
                        loc = loc + Integer.parseInt(op2);
                        i++;
                  }
            }
            else if (check(m) == 3) {
                  if(input[i][0] == null) {
                        int temp;
                        op1 = input[i][2];
                        op2 = input[i][3];
                        temp = comps(op1, s, sn);
                        if (temp !=99) {
                              s[temp] = new symtab(temp, op1, loc);
                        }
                        loc++;
                        i++;
                  }
```

```
                        else {
                              int temp;
                              op1 = input[i][0];
                              s[sn] = new symtab(sn, op1, loc);
                              sn++;

                              op1 = input[i][2];
                              op2 = input[i][3];
                              temp = comps(op1, s, sn);
                              if (temp != 99) {
                                    s[temp] = new symtab(temp, op1, loc);
                              }
                              loc++;
                              i++;
                        }
                  }
                  else if (check(m) == 4) {
                        if(lnc != ln) {
                              p[pn] = new pooltab(pn,lnc);
                              pn++;
                        }
                        while (lnc != ln) {
                              l[lnc].setaddr(loc);
                              lnc++;
                              loc++;
                        }
                        i++;
                  }
                  m = input [i][1];
            }
            if (lnc != ln) {
                  p[pn] = new pooltab(pn,lnc);
                  pn++;
            }
            while (lnc != ln) {
                  l[lnc].setaddr(loc);
                  lnc++;
                  loc++;
            }
            System.out.println("Symbol Table\nIndex\tSymbol\tAddress\n");

            for (i=0; i<sn; i++) {
                  System.out.println(s[i].index + "\t" + s[i].name + "\t"
+ s[i].addr);
            }
            System.out.println("\nLiteral
Table\nIndex\tLiteral\tAddress\n");
            for (i=0; i<ln; i++) {
                  System.out.println(l[i].index + "\t" + l[i].name + "\t"
+ l[i].addr);
            }
            System.out.println("\nPool Table\nPool Index\tLiteral
Index\n");
            for (i=0; i<pn; i++) {
                System.out.println("\t" + p[i].p_index + "\t\t" +
p[i].l_index);
            }
            System.out.println("\n\nIntermediate Code\n");
            i=0;
            m = input[i][1];
            op1 = input[i][2];
            op2 = input[i][3];

            int point = 0, in1, in2, j=0;

            System.out.println(ic(m) + ic(op1));
            while (!m.equals("END")) {
                  if (check(m) == 1) {
                        System.out.println(ic(m) + ic(op1));
                        if (comp(op2,s,sn) == 0 && comps(op2, s, sn) ==
99) {
                              System.out.println(ic(op2));
                        }
                        else if (comp(op2, s, sn) == 2) {
                              int temp;
                              temp = compl(op2, l, ln, j);
                              System.out.println("(L," + temp + ")");
                              j++;
                        }
                        else if (comp(op2, s, sn) != 1) {
                              int temp;
                              temp = comps(op2, s, sn);
                              System.out.println("S," + temp + ")");
                        }
                  }
                  else if (check(m) == 2 || check(m) == 3) {
                        System.out.println(ic(m) + ic(op2));
                        /*if (comp(op1, s, sn) != 1) {
                              int temp;
                              temp = comps(op1, s, sn);
                              System.out.println("(S," + temp + ")");
                        }*/
                  }
                  else if (check(m) == 4) {
                        if (point+1 != pn) {
                              in1 = p[point+1].l_index - p[point].l_index;

                              in2 = p[point].l_index;
                              point++;

                              while (in1>0) {
                                    System.out.println(ic(m) +
ic(l[in2].name));
                                    in2++;
                                    in1--;
                                    System.out.println("\n");
                              }
                        }
                        else {
                              in2 = p[point].l_index;
                              while (in2 != ln) {
                                    System.out.println(ic(m) +
ic(l[in2].name));
                                    in2++;
                                    System.out.println("\n");
                              }
                        }
                  }
                  i++;
                  m = input[i][1];
                  op1 = input[i][2];
                  op2 = input[i][3];
                  System.out.println("\n");
            }
            System.out.println(ic(m));
            m = "LTORG";
            if (point+1 != pn) {
                  in1 = p[point+1].l_index - p[point].l_index;
                  in2 = p[point].l_index;
                  point++;
                  while (in1 > 0) {
                        System.out.println(ic(m) + ic(l[in2].name));
                        in2++;
                        in1--;
                  }
            }
            else {
                  in2 = p[point].l_index;
                  while (in2 != ln) {
                        System.out.println(ic(m) + ic(l[in2].name));
                        in2++;
                  }
            }
      }

      static int check(String m) {
            if (m.equals("MOVER") || m.equals("ADD")) {
                  return 1;
            }
            else if (m.equals("DS")) {
                  return 2;
            }
            else if (m.equals("DC")) {
                  return 3;
            }
            else if (m.equals("LTORG")) {
                  return 4;
            }
            return -1;
      }

      static int comp(String m, symtab s[], int sn) {
```

```
				if (m.equals("AREG") || m.equals("BREG") ||
m.equals("CREG"))
					return 0;
				else if (m.toCharArray()[0] == '=')
					return 2;
				else if (comps(m, s, sn) == 99)
					return 1;
				else
					return 0;
		}

		static int compl(String m, littab l[], int ln, int j) {
			int i;
			for (i=j; i<ln; i++) {
				if (m.equals(l[i].name))
					return l[i].index;
			}
			return 99;
		}

		static int comps(String m, symtab s[], int sn) {
			int i;
			for (i=0; i<sn; i++) {
				if (m.equals(s[i].name))
					return s[i].index;
			}
			return 99;
		}

		static String ic(String m) {
			if (m == "START")
				return "(AD, 01)";
			else if ( m == "END")
				return "(AD, 02)";
			else if ( m == "ORIGIN")
				return "(AD, 03)";
			else if ( m == "EQU")
				return "(AD, 04)";
			else if ( m == "LTORG")
				return "(DL, 02)";
			else if ( m == "ADD")
				return "(IS, 01)";
			else if ( m == "SUB")
				return "(IS, 02)";
			else if ( m == "MOVER")
				return "(IS, 04)";
			else if ( m == "MOVEM")
				return "(AD, 05)";
			else if ( m == "AREG")
				return "(RG, 01)";
			else if ( m == "BREG")
				return "(RG, 02)";
			else if ( m == "CREG")
				return "(RG, 03)";
```

```
            else if ( m == "DS")
                  return "(DL, 01)";
            else if ( m == "DC")
                  return "(DL, 02)";
            else if (m.toCharArray()[0] == '=')
                  return ( "(C," + m.toCharArray()[2] + ")" );
            else {
                  return ("(C," + m + ")");

            }
      }
}
```