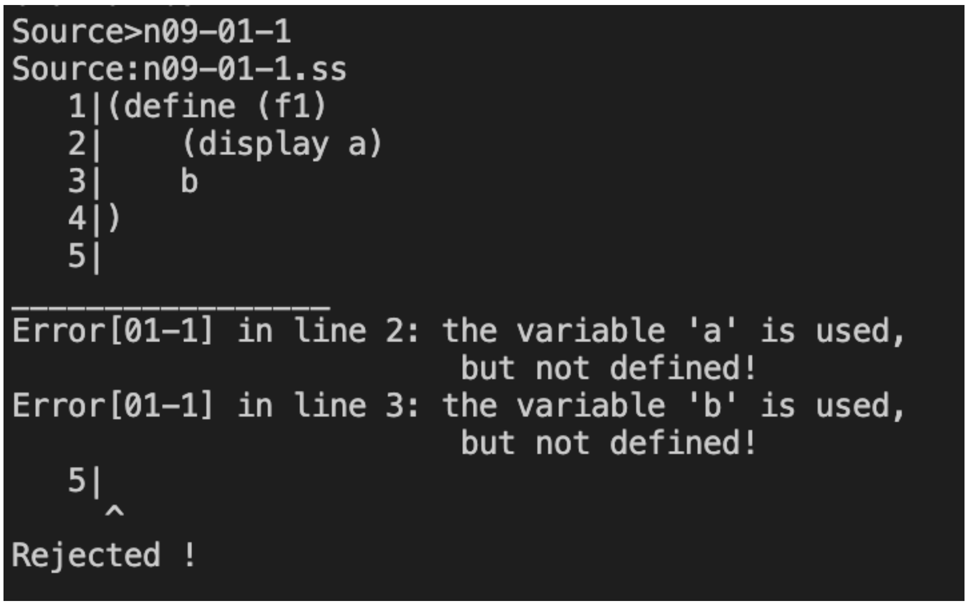
Студент: Епанешников В. С.
Группа: М8О-206Б-19
Номер по списку: 9

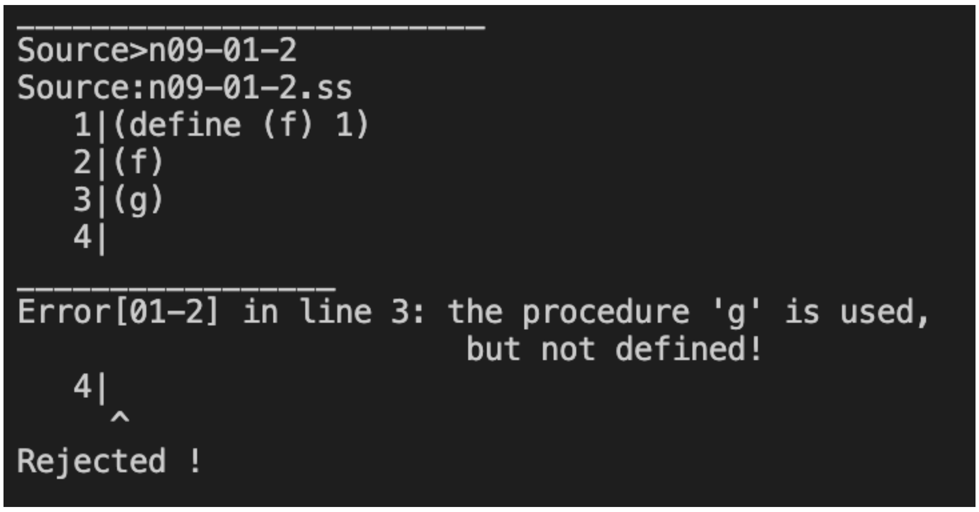
«СИСТЕМЫ ПРОГРАММИРОВАНИЯ» Курсовая работа 2021. Часть 2.

Перечень документов в отчете. Вариант грамматики:n09

Скриншоты всех тестов, упорядоченные по номерам продукций и сообщений.
>
P01: S -> PROG

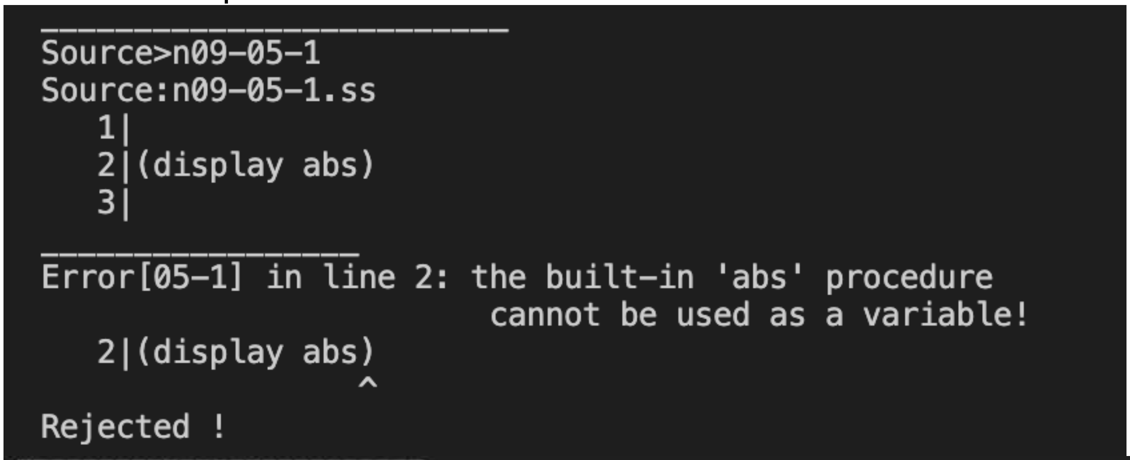


```
Source>n09-01-1
Source:n09-01-1.ss
   1|(define (f1)
   2|    (display a)
   3|    b
   4|)
   5|
_________________
Error[01-1] in line 2: the variable 'a' is used,
                        but not defined!
Error[01-1] in line 3: the variable 'b' is used,
                        but not defined!
   5|
     ^
Rejected !
```

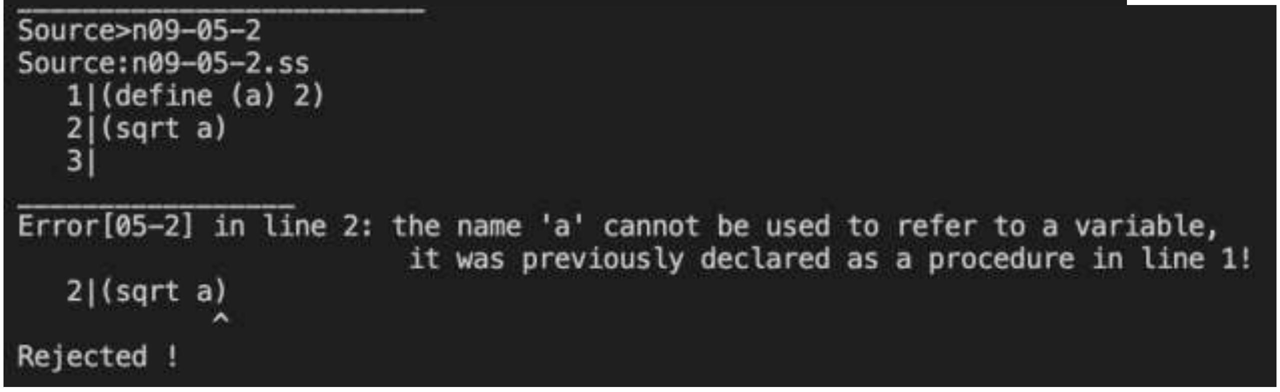


```
__________________________
Source>n09-01-2
Source:n09-01-2.ss
   1|(define (f) 1)
   2|(f)
   3|(g)
   4|

_________________
Error[01-2] in line 3: the procedure 'g' is used,
                        but not defined!
   4|
     ^
Rejected !
```

P05: E -> $id

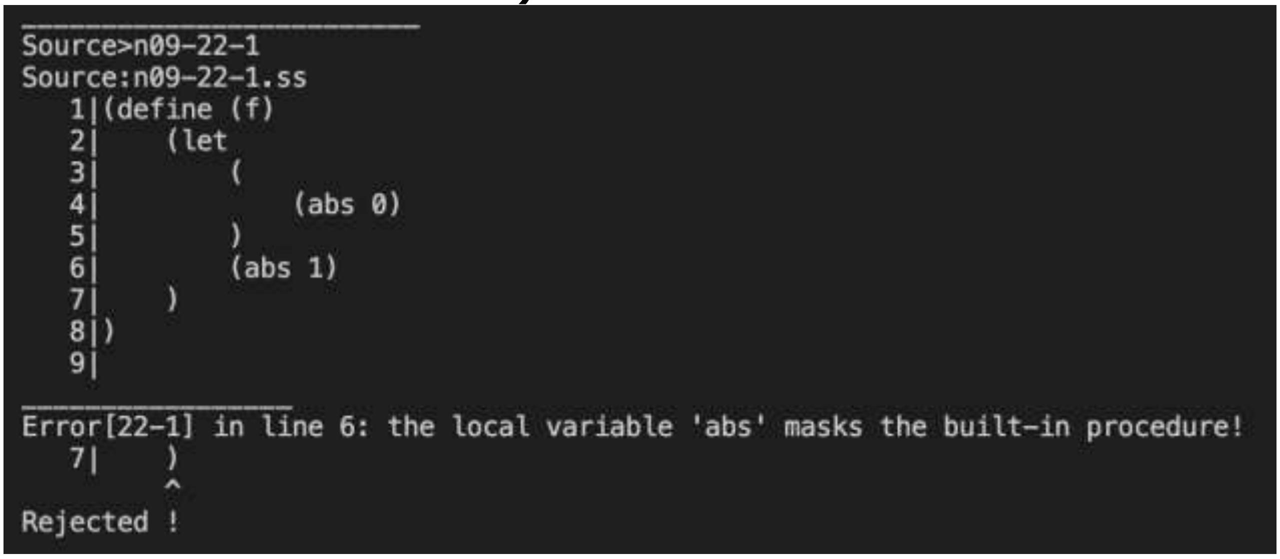


```
__________________________
Source>n09-05-1
Source:n09-05-1.ss
   1|
   2|(display abs)
   3|

_________________
Error[05-1] in line 2: the built-in 'abs' procedure
                        cannot be used as a variable!
   2|(display abs)
                 ^
Rejected !
```

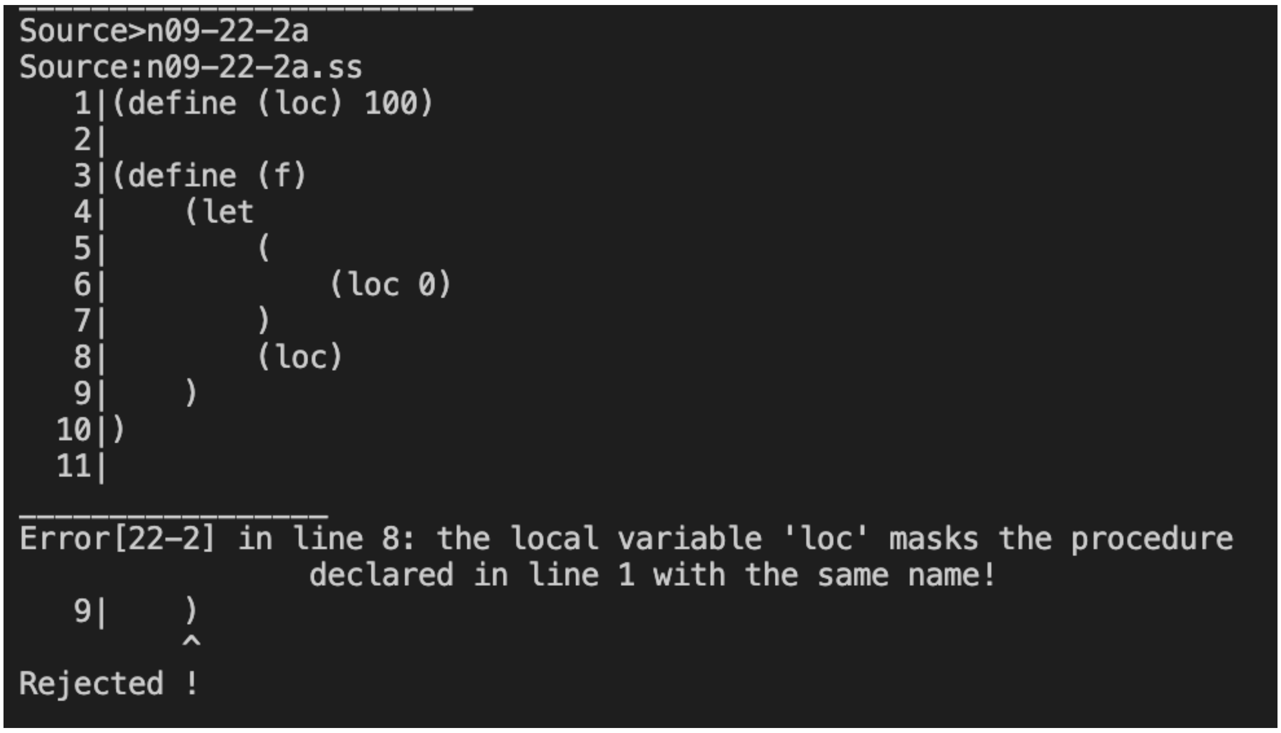


```
__________________________
Source>n09-05-2
Source:n09-05-2.ss
   1|(define (a) 2)
   2|(sqrt a)
   3|

_________________
Error[05-2] in line 2: the name 'a' cannot be used to refer to a variable,
                        it was previously declared as a procedure in line 1!
   2|(sqrt a)
           ^
Rejected !
```

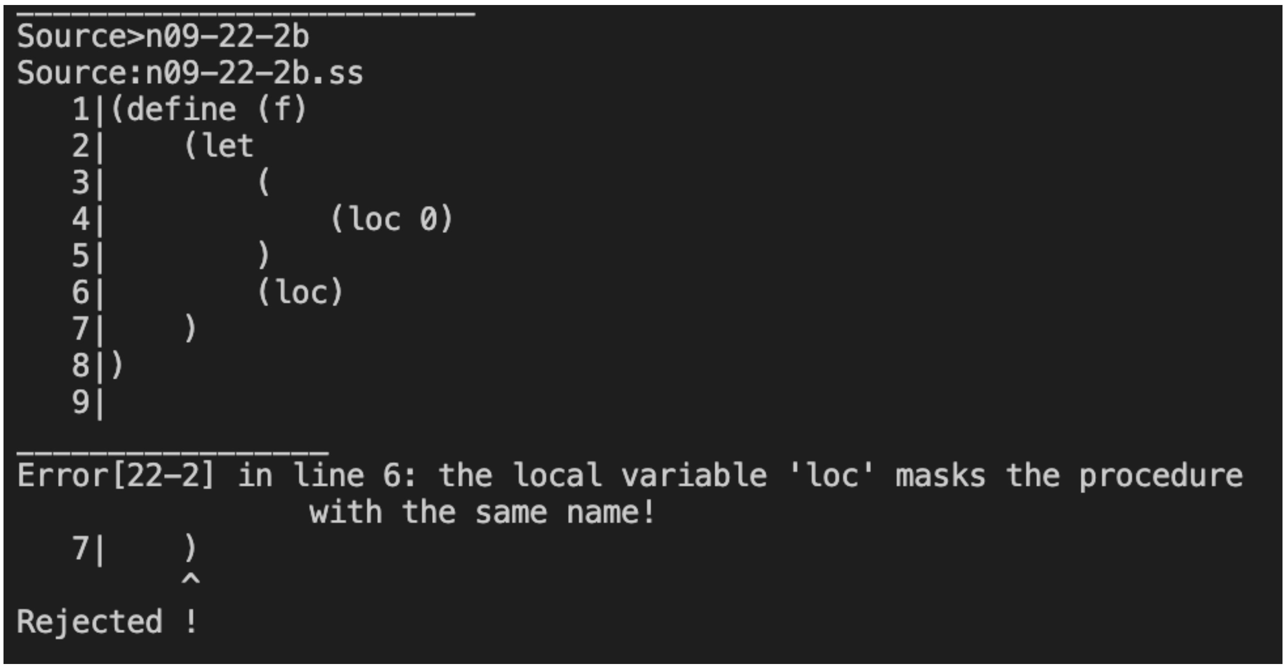
P22: CPROC -> HCPROC)



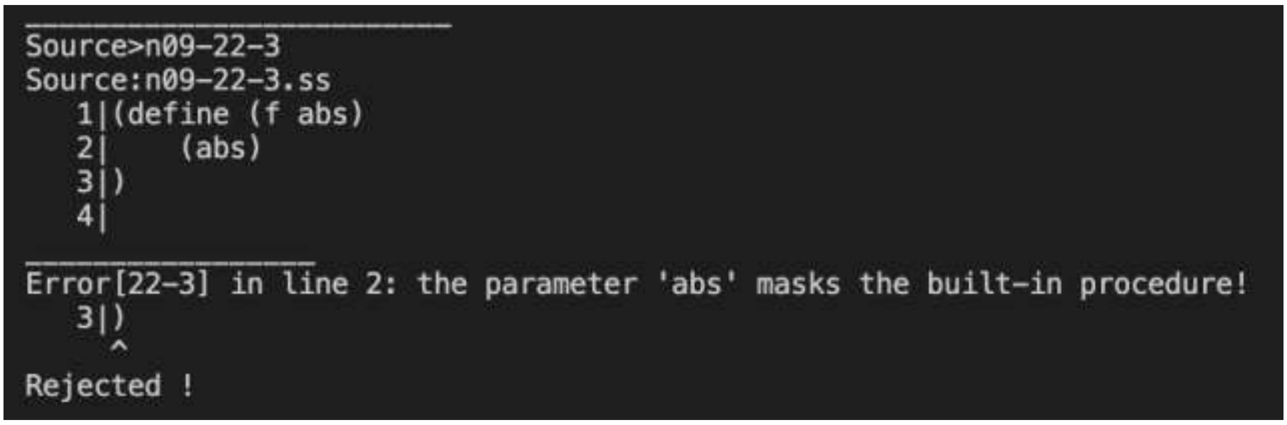
```
__________________________
Source>n09-22-1
Source:n09-22-1.ss
   1|(define (f)
   2|    (let
   3|        (
   4|            (abs 0)
   5|        )
   6|        (abs 1)
   7|    )
   8|)
   9|
_________________
Error[22-1] in line 6: the local variable 'abs' masks the built-in procedure!
   7|    )
         ^
Rejected !
```



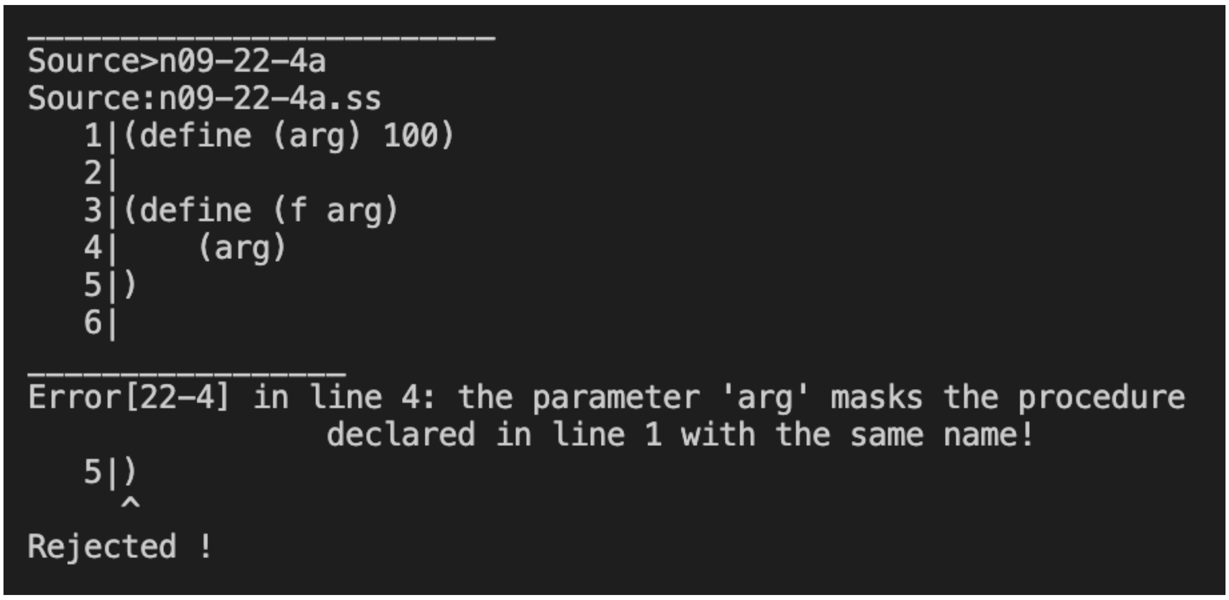
```
_________________________
Source>n09-22-2a
Source:n09-22-2a.ss
   1|(define (loc) 100)
   2|
   3|(define (f)
   4|    (let
   5|        (
   6|            (loc 0)
   7|        )
   8|        (loc)
   9|    )
  10|)
  11|
_________________
Error[22-2] in line 8: the local variable 'loc' masks the procedure
                declared in line 1 with the same name!
   9|    )
         ^
Rejected !
```



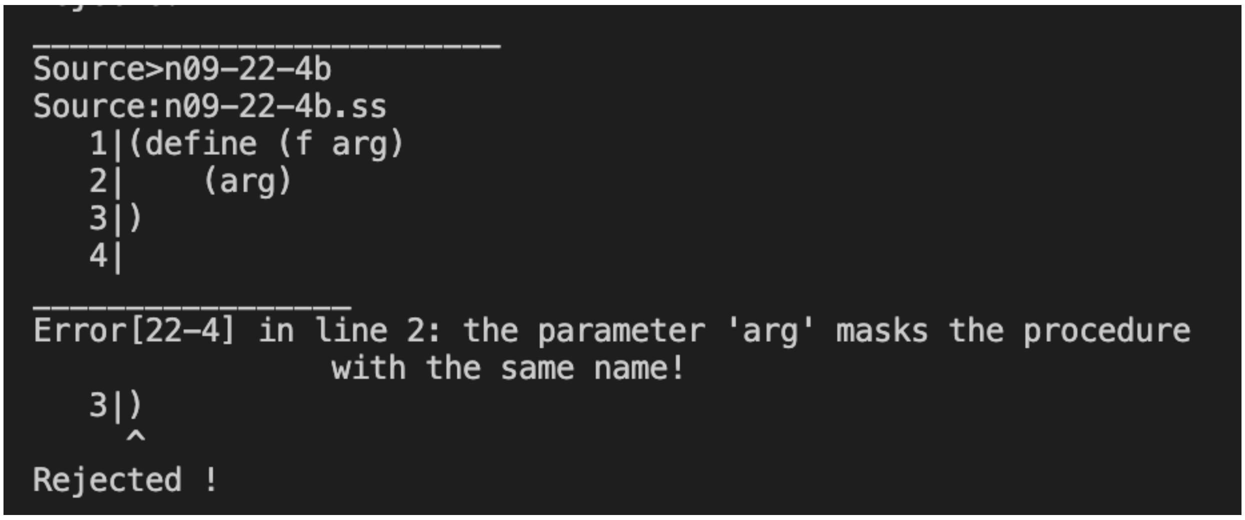
```
--------------------------
Source>n09-22-2b
Source:n09-22-2b.ss
   1|(define (f)
   2|    (let
   3|        (
   4|            (loc 0)
   5|        )
   6|        (loc)
   7|    )
   8|)
   9|
_________________
Error[22-2] in line 6: the local variable 'loc' masks the procedure
                with the same name!
   7|    )
         ^
Rejected !
```



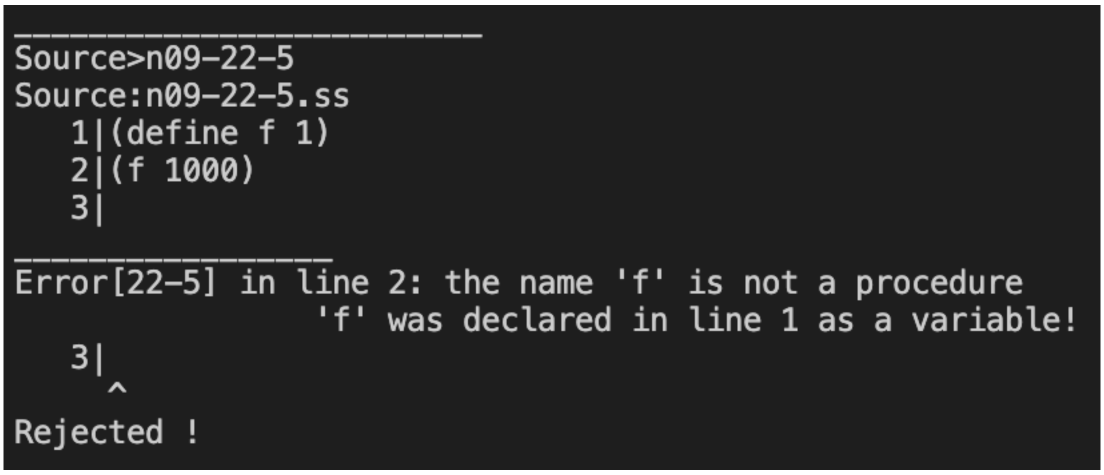
```
--------------------------
Source>n09-22-3
Source:n09-22-3.ss
   1|(define (f abs)
   2|    (abs)
   3|)
   4|
_________________
Error[22-3] in line 2: the parameter 'abs' masks the built-in procedure!
   3|)
     ^
Rejected !
```



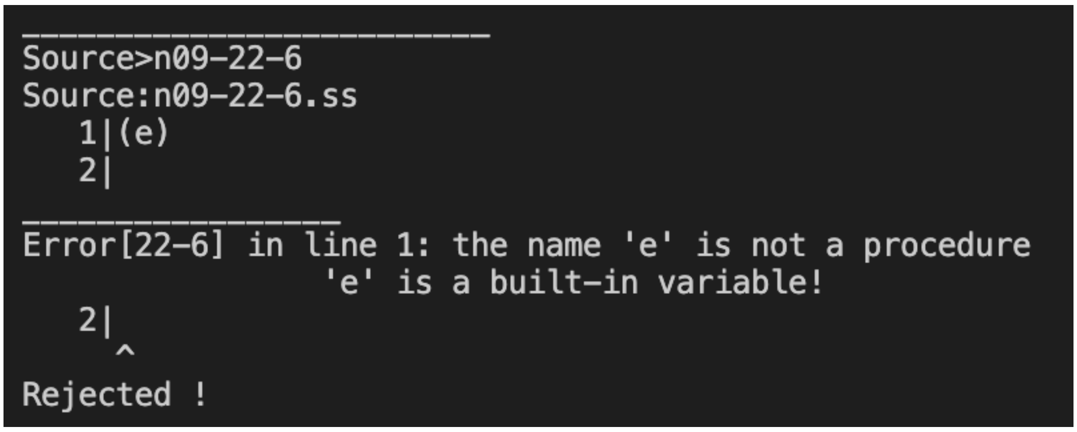
```
--------------------------
Source>n09-22-4a
Source:n09-22-4a.ss
   1|(define (arg) 100)
   2|
   3|(define (f arg)
   4|    (arg)
   5|)
   6|
_________________
Error[22-4] in line 4: the parameter 'arg' masks the procedure
                declared in line 1 with the same name!
   5|)
     ^
Rejected !
```



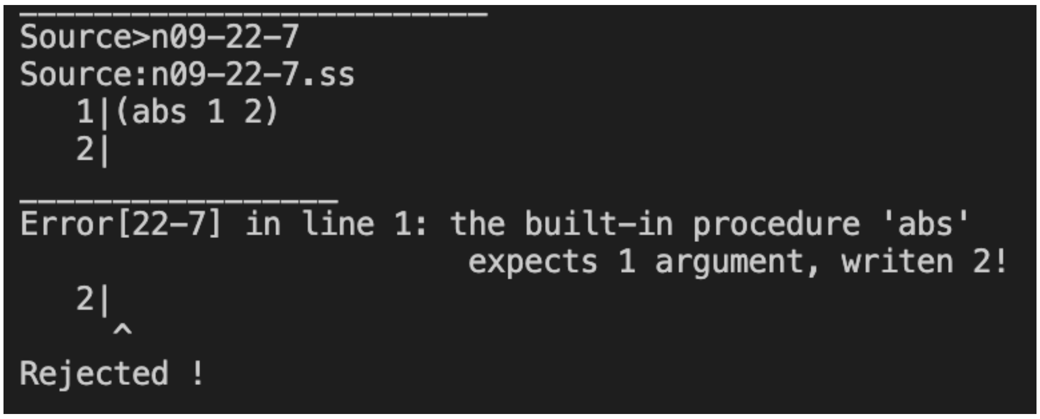
```
 _________________________
 Source>n09-22-4b
 Source:n09-22-4b.ss
    1|(define (f arg)
    2|    (arg)
    3|)
    4|
 _________________
 Error[22-4] in line 2: the parameter 'arg' masks the procedure
                 with the same name!
    3|)
      ^
 Rejected !
```



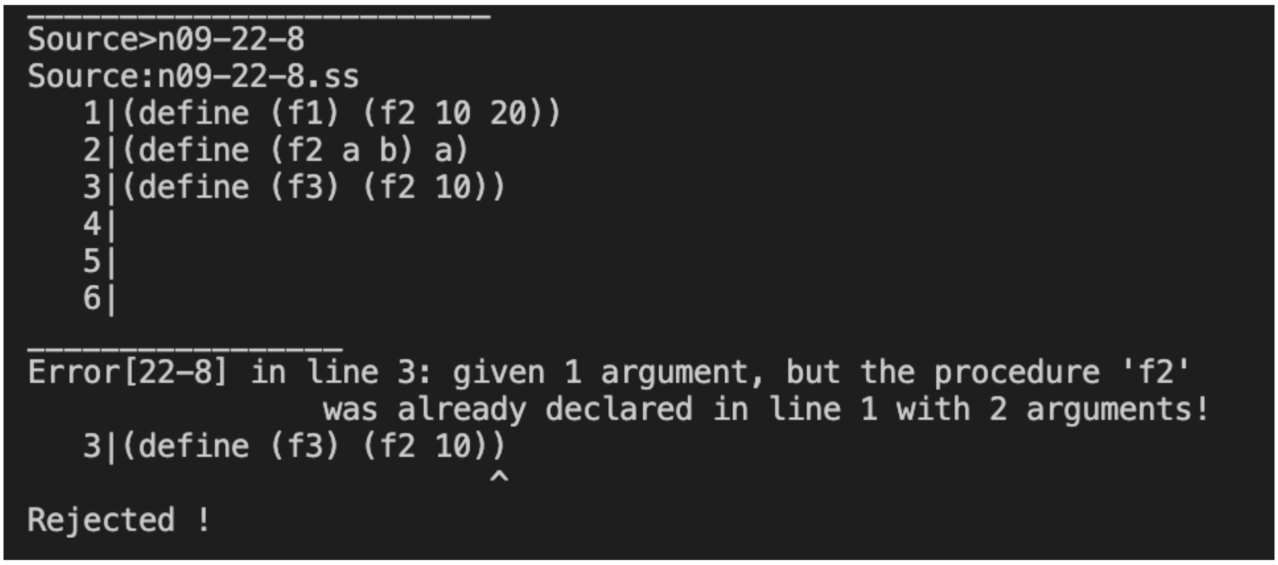
```
_________________________
Source>n09-22-5
Source:n09-22-5.ss
   1|(define f 1)
   2|(f 1000)
   3|
_________________
Error[22-5] in line 2: the name 'f' is not a procedure
                'f' was declared in line 1 as a variable!
   3|
     ^
Rejected !
```



```
_________________________
Source>n09-22-6
Source:n09-22-6.ss
   1|(e)
   2|
_________________
Error[22-6] in line 1: the name 'e' is not a procedure
                'e' is a built-in variable!
   2|
     ^
Rejected !
```

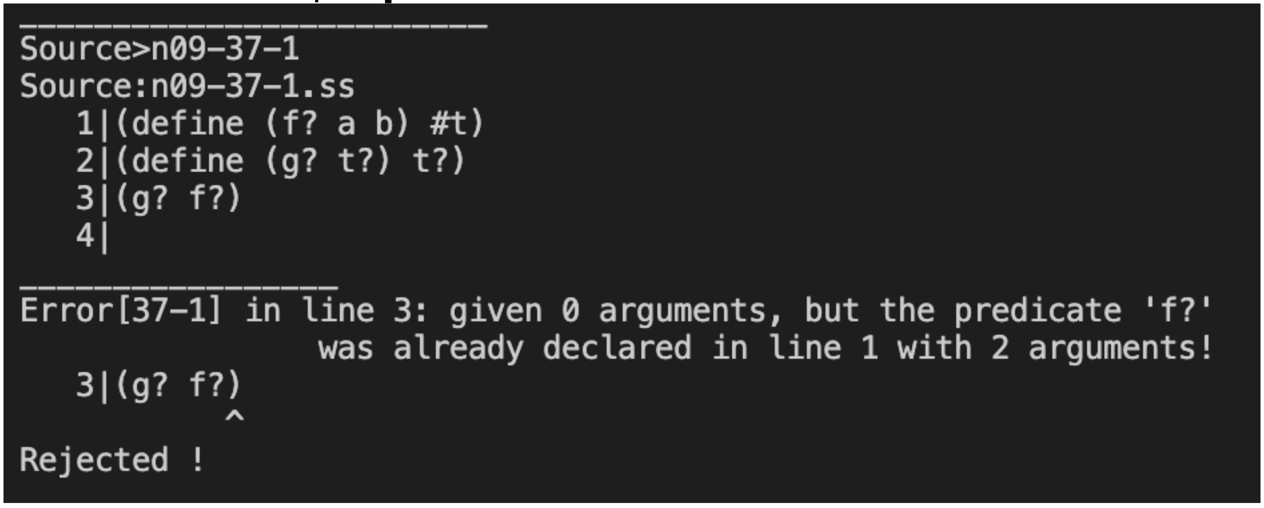


```
_________________________
Source>n09-22-7
Source:n09-22-7.ss
   1|(abs 1 2)
   2|
_________________
Error[22-7] in line 1: the built-in procedure 'abs'
                        expects 1 argument, writen 2!
   2|
     ^
Rejected !
```



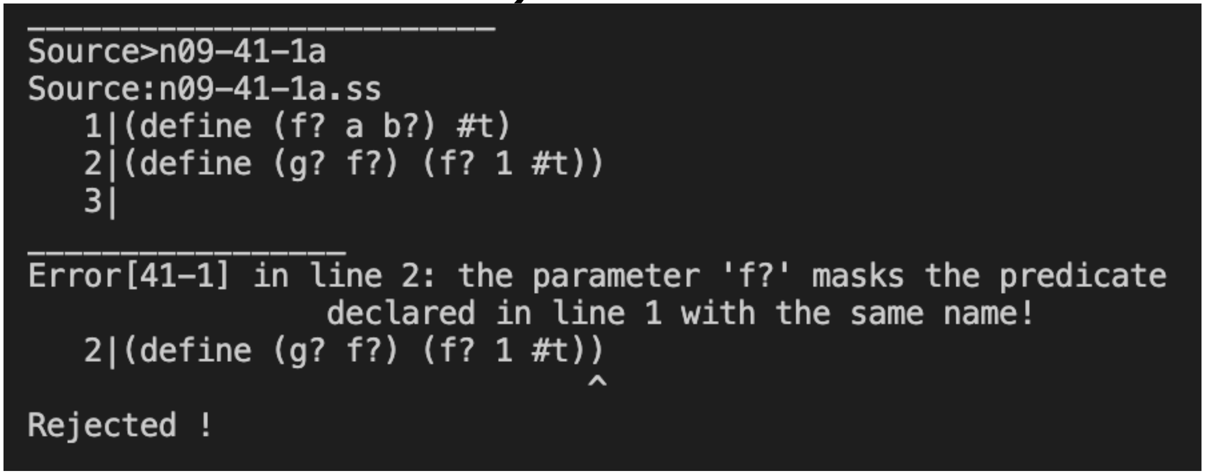
```
-------------------------
Source>n09-22-8
Source:n09-22-8.ss
   1|(define (f1) (f2 10 20))
   2|(define (f2 a b) a)
   3|(define (f3) (f2 10))
   4|
   5|
   6|
_________________
Error[22-8] in line 3: given 1 argument, but the procedure 'f2'
                was already declared in line 1 with 2 arguments!
   3|(define (f3) (f2 10))
                          ^
Rejected !
```

P37: BOOL -> $idq

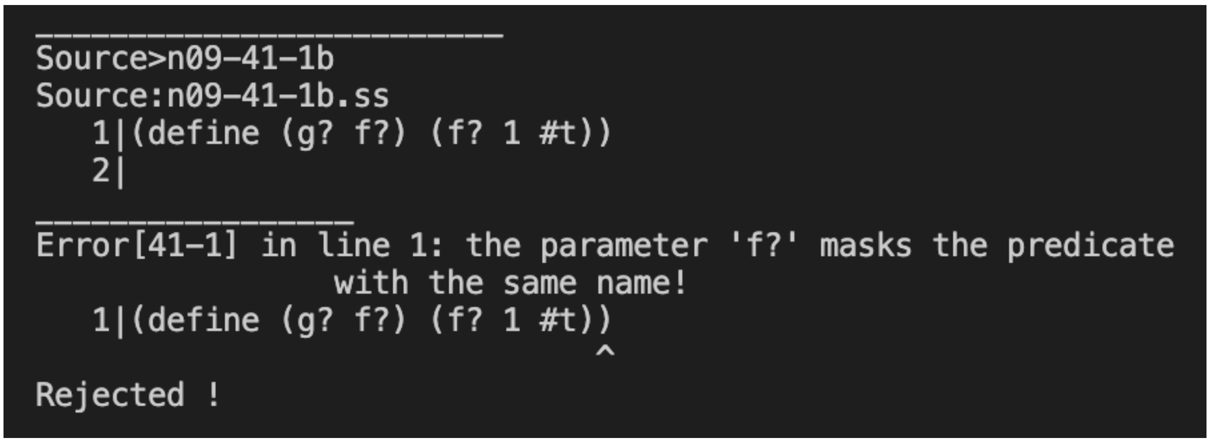


```
-------------------------
Source>n09-37-1
Source:n09-37-1.ss
   1|(define (f? a b) #t)
   2|(define (g? t?) t?)
   3|(g? f?)
   4|
_________________
Error[37-1] in line 3: given 0 arguments, but the predicate 'f?'
                was already declared in line 1 with 2 arguments!
   3|(g? f?)
           ^
Rejected !
```

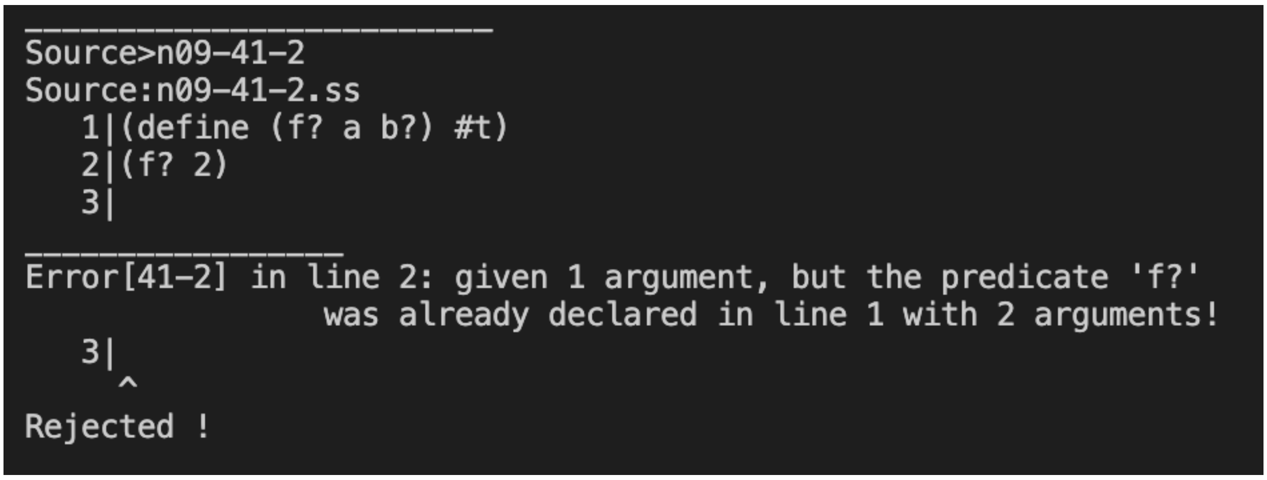
P41: CPRED -> HCPRED)



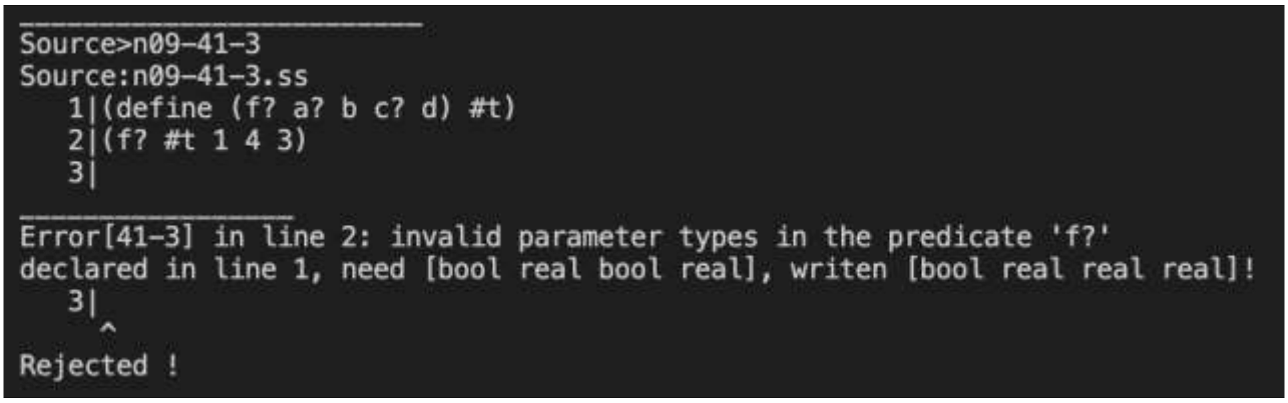
```
-------------------------
Source>n09-41-1a
Source:n09-41-1a.ss
   1|(define (f? a b?) #t)
   2|(define (g? f?) (f? 1 #t))
   3|
_________________
Error[41-1] in line 2: the parameter 'f?' masks the predicate
                declared in line 1 with the same name!
   2|(define (g? f?) (f? 1 #t))
                              ^
Rejected !
```



```
_________________________
Source>n09-41-1b
Source:n09-41-1b.ss
   1|(define (g? f?) (f? 1 #t))
   2|
_________________
Error[41-1] in line 1: the parameter 'f?' masks the predicate
                with the same name!
   1|(define (g? f?) (f? 1 #t))
                              ^
Rejected !
```

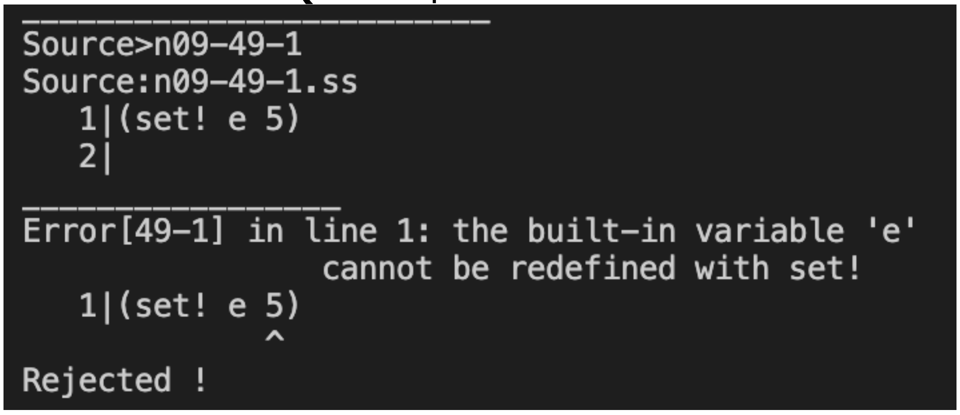


```
_________________________
Source>n09-41-2
Source:n09-41-2.ss
   1|(define (f? a b?) #t)
   2|(f? 2)
   3|
_________________
Error[41-2] in line 2: given 1 argument, but the predicate 'f?'
                was already declared in line 1 with 2 arguments!
   3|
     ^
Rejected !
```

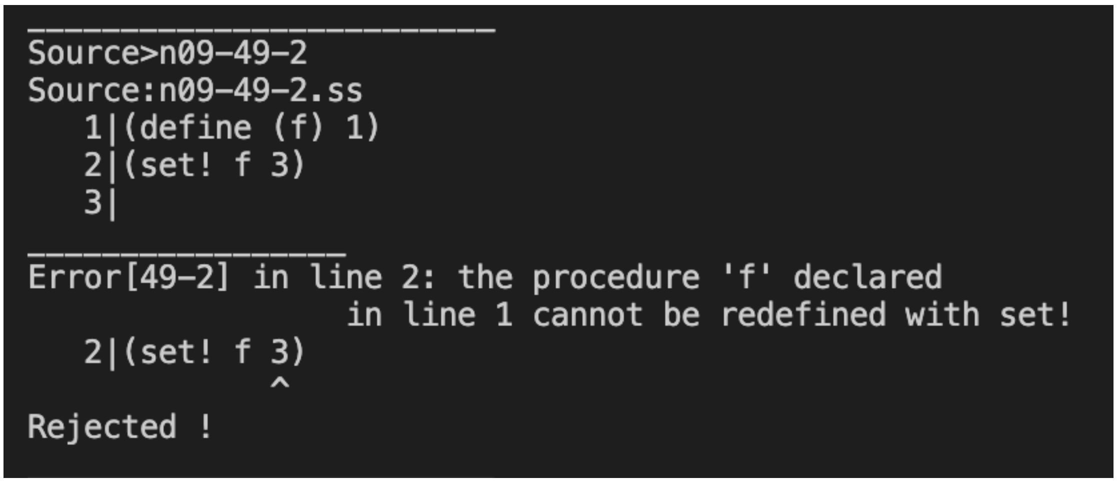


```
_________________________
Source>n09-41-3
Source:n09-41-3.ss
   1|(define (f? a? b c? d) #t)
   2|(f? #t 1 4 3)
   3|
_________________
Error[41-3] in line 2: invalid parameter types in the predicate 'f?'
declared in line 1, need [bool real bool real], writen [bool real real real]!
   3|
     ^
Rejected !
```

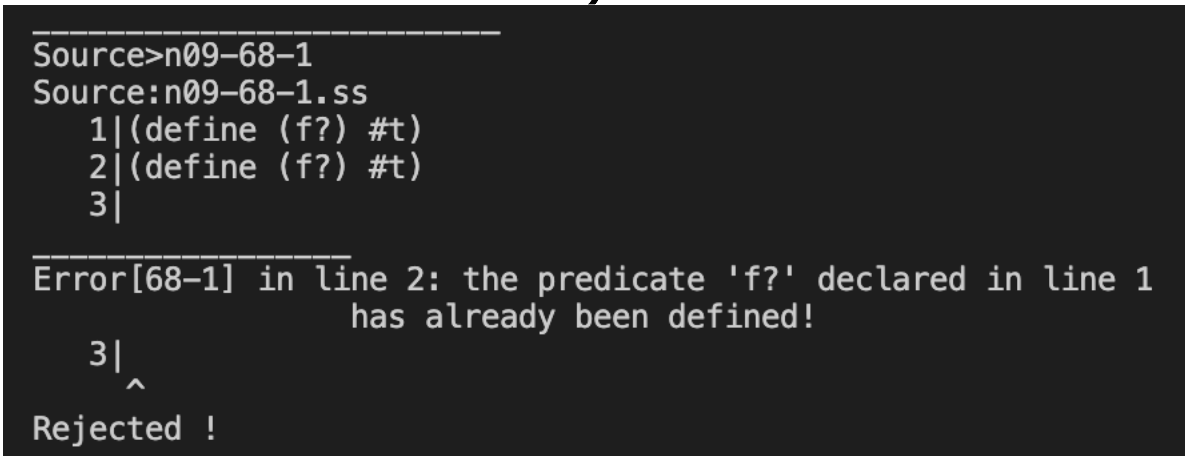
P49: HSET -> (set! $id

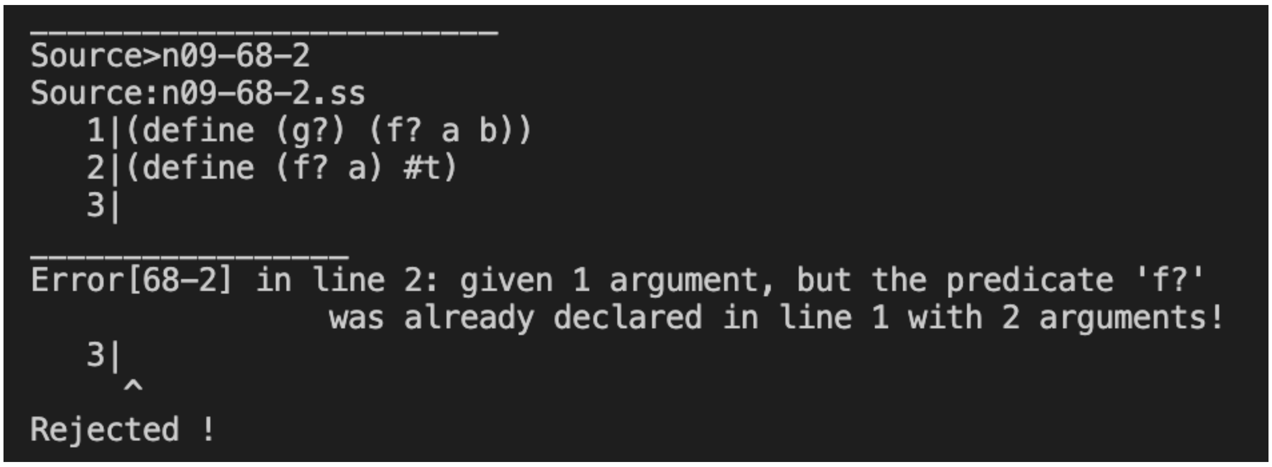


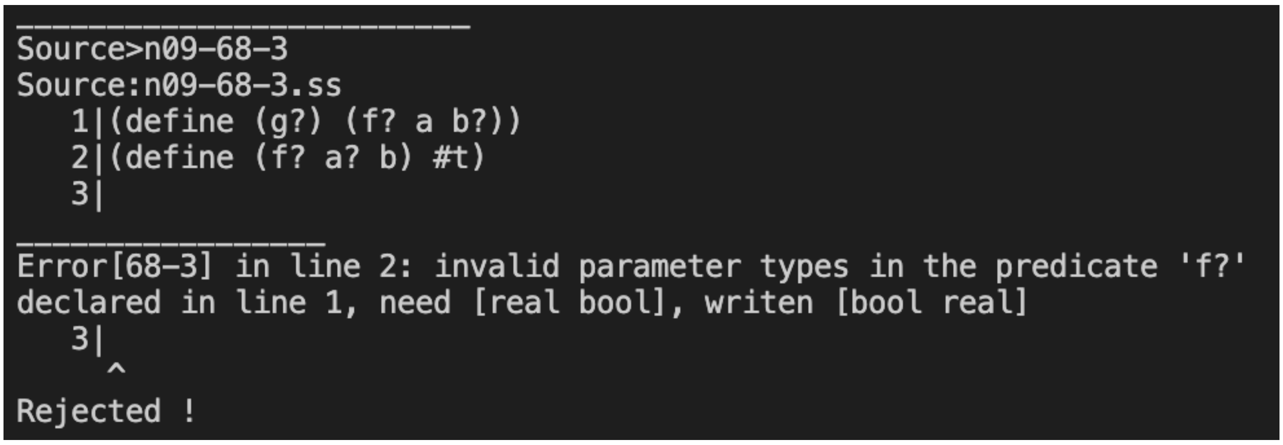
```
_________________________
Source>n09-49-1
Source:n09-49-1.ss
   1|(set! e 5)
   2|
_________________
Error[49-1] in line 1: the built-in variable 'e'
                cannot be redefined with set!
   1|(set! e 5)
             ^
Rejected !
```


```
_________________________
Source>n09-49-2
Source:n09-49-2.ss
   1|(define (f) 1)
   2|(set! f 3)
   3|
_________________
Error[49-2] in line 2: the procedure 'f' declared
                 in line 1 cannot be redefined with set!
   2|(set! f 3)
             ^
Rejected !
```

P68: PRED -> HPRED BOOL)

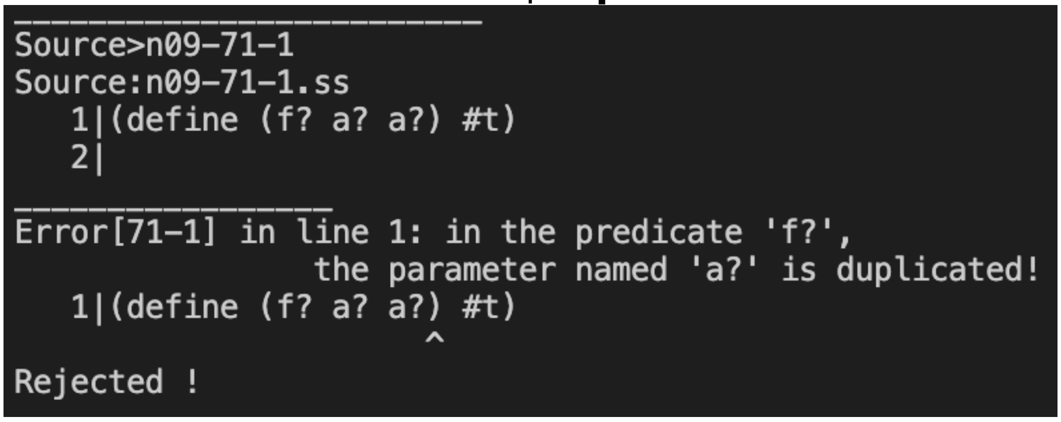

```
_________________________
Source>n09-68-1
Source:n09-68-1.ss
   1|(define (f?) #t)
   2|(define (f?) #t)
   3|
_________________
Error[68-1] in line 2: the predicate 'f?' declared in line 1
                 has already been defined!
   3|
     ^
Rejected !
```


```
_________________________
Source>n09-68-2
Source:n09-68-2.ss
   1|(define (g?) (f? a b))
   2|(define (f? a) #t)
   3|
_________________
Error[68-2] in line 2: given 1 argument, but the predicate 'f?'
                was already declared in line 1 with 2 arguments!
   3|
     ^
Rejected !
```

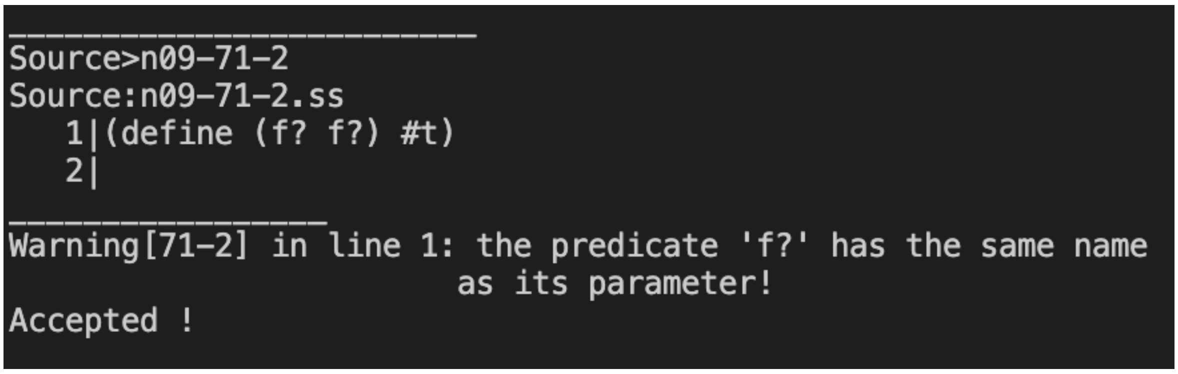


```
_________________________
Source>n09-68-3
Source:n09-68-3.ss
   1|(define (g?) (f? a b?))
   2|(define (f? a? b) #t)
   3|
_________________
Error[68-3] in line 2: invalid parameter types in the predicate 'f?'
declared in line 1, need [real bool], writen [bool real]
   3|
     ^
Rejected !
```

P71: PDPAR -> PRDPAR $idq

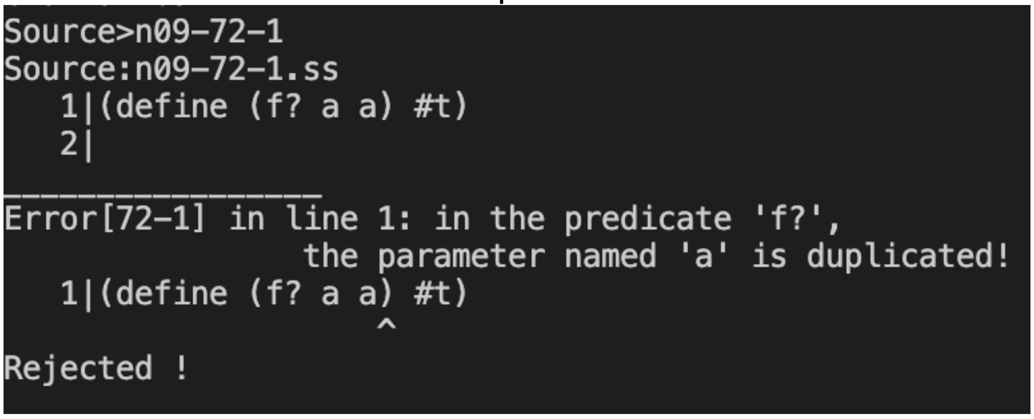


```
_________________________
Source>n09-71-1
Source:n09-71-1.ss
   1|(define (f? a? a?) #t)
   2|
_________________
Error[71-1] in line 1: in the predicate 'f?',
                the parameter named 'a?' is duplicated!
   1|(define (f? a? a?) #t)
                       ^
Rejected !
```



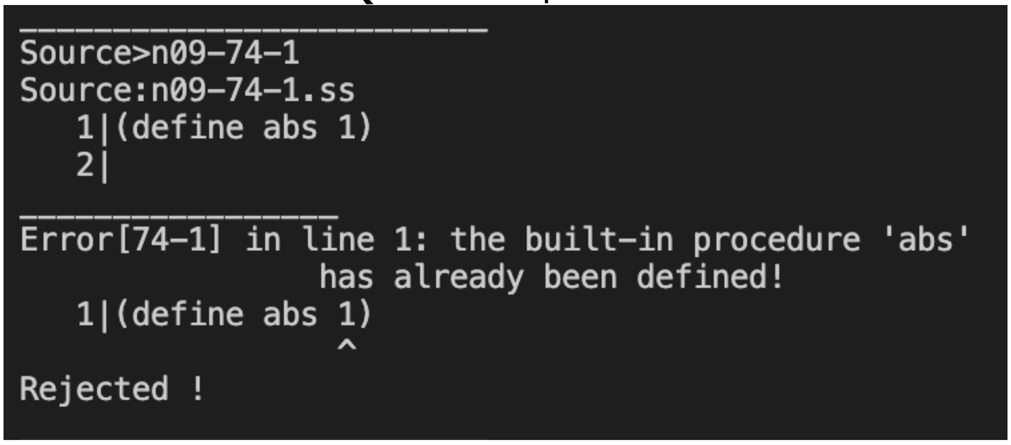
```
_________________________
Source>n09-71-2
Source:n09-71-2.ss
   1|(define (f? f?) #t)
   2|
_________________
Warning[71-2] in line 1: the predicate 'f?' has the same name
                         as its parameter!
Accepted !
```

P72: PDPAR -> PDPAR $id

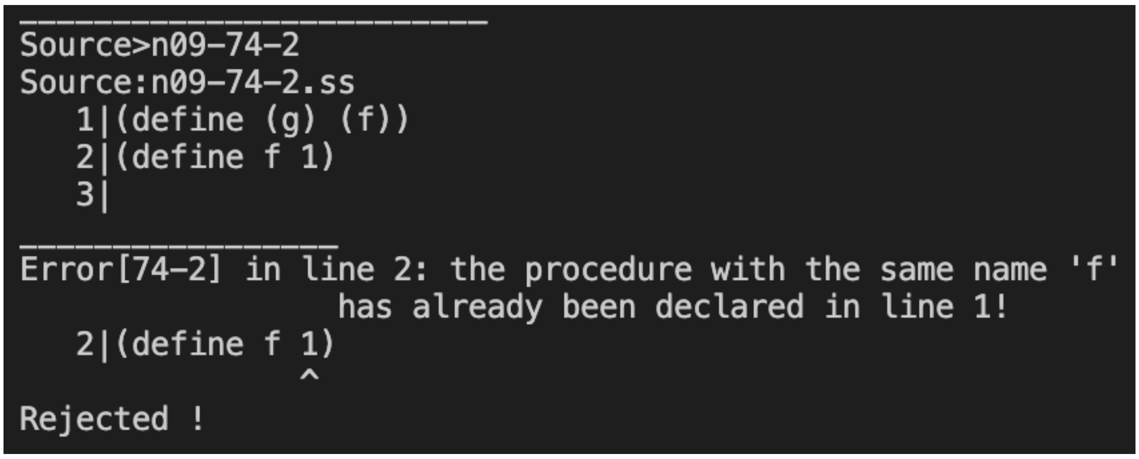


```
Source>n09-72-1
Source:n09-72-1.ss
   1|(define (f? a a) #t)
   2|
_________________
Error[72-1] in line 1: in the predicate 'f?',
                the parameter named 'a' is duplicated!
   1|(define (f? a a) #t)
                     ^
Rejected !
```

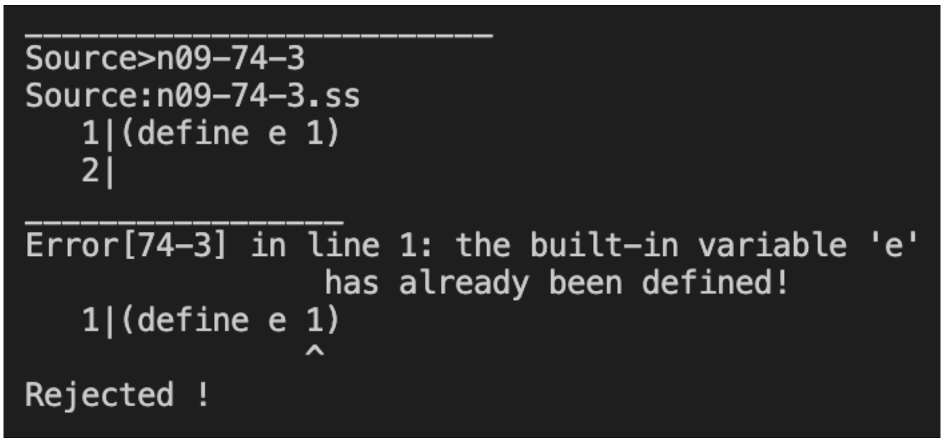
P74: VARDCL -> (define $id



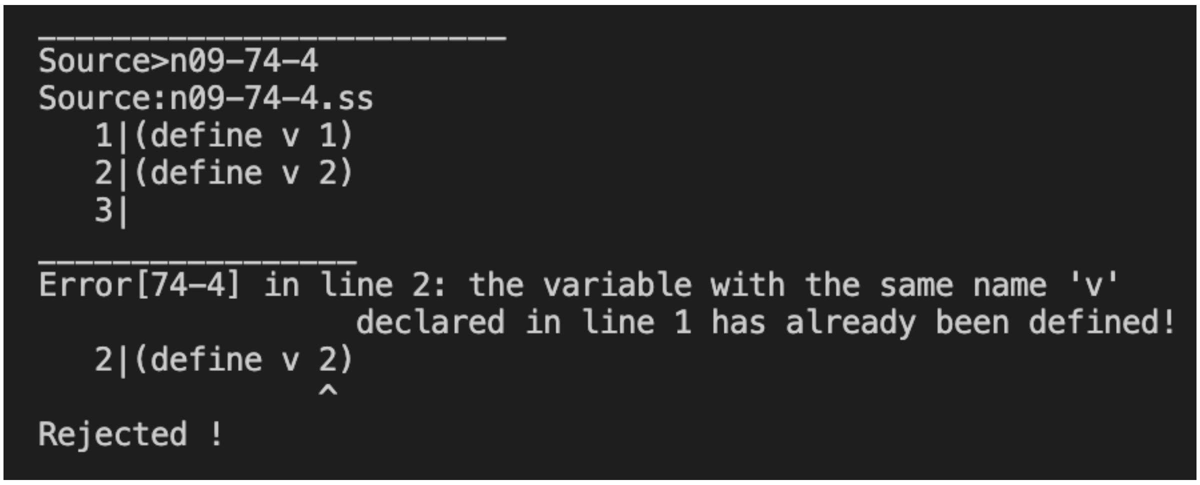
```
_________________________
Source>n09-74-1
Source:n09-74-1.ss
   1|(define abs 1)
   2|
_________________
Error[74-1] in line 1: the built-in procedure 'abs'
                has already been defined!
   1|(define abs 1)
                 ^
Rejected !
```



```
_________________________
Source>n09-74-2
Source:n09-74-2.ss
   1|(define (g) (f))
   2|(define f 1)
   3|
_________________
Error[74-2] in line 2: the procedure with the same name 'f'
                 has already been declared in line 1!
   2|(define f 1)
               ^
Rejected !
```

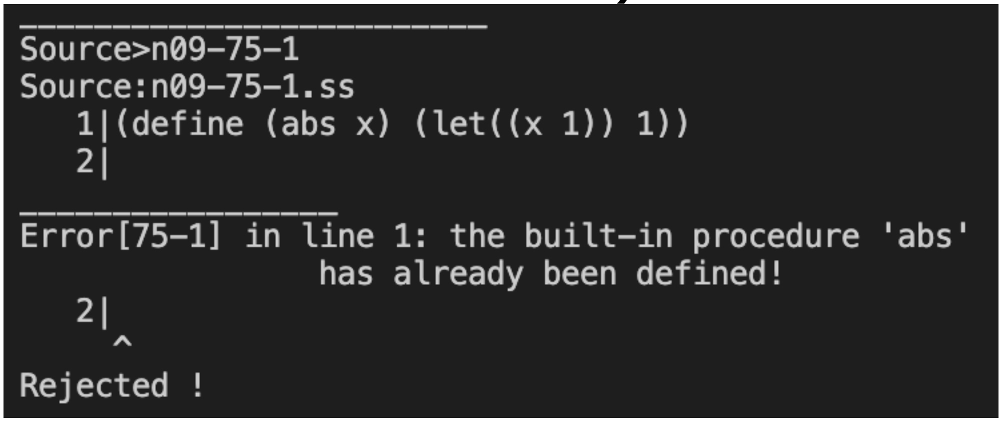


```
_________________________
Source>n09-74-3
Source:n09-74-3.ss
   1|(define e 1)
   2|
_________________
Error[74-3] in line 1: the built-in variable 'e'
                has already been defined!
   1|(define e 1)
               ^
Rejected !
```

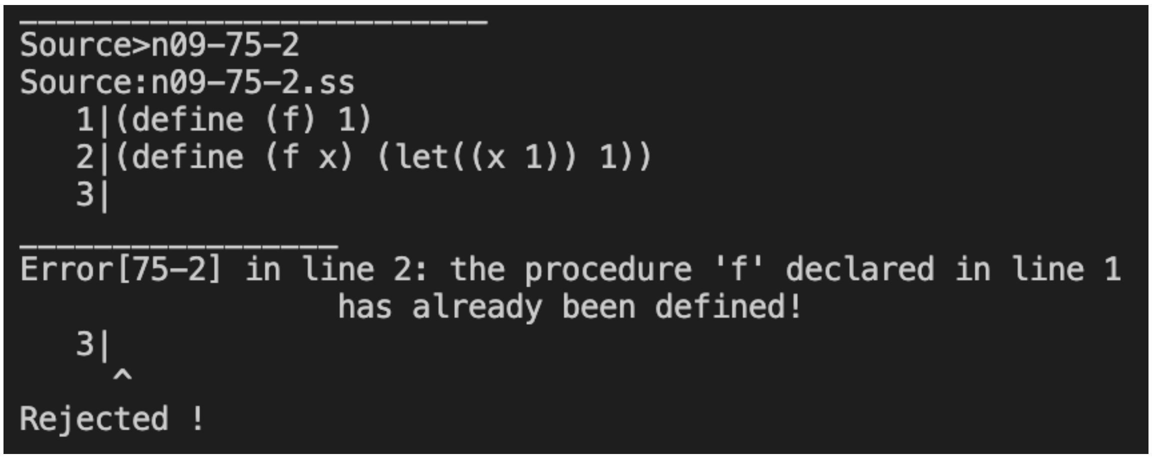


```
_________________________
Source>n09-74-4
Source:n09-74-4.ss
   1|(define v 1)
   2|(define v 2)
   3|
_________________
Error[74-4] in line 2: the variable with the same name 'v'
                 declared in line 1 has already been defined!
   2|(define v 2)
              ^
Rejected !
```

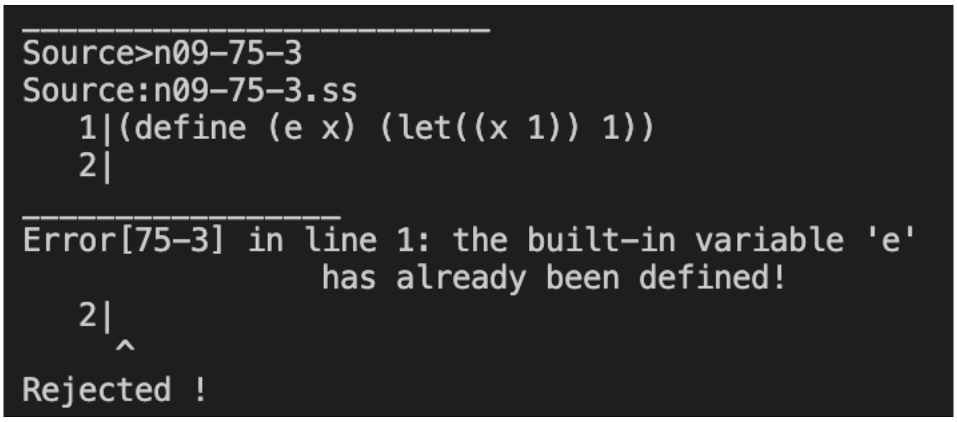
P75: PROC -> HPROC BLOCK)



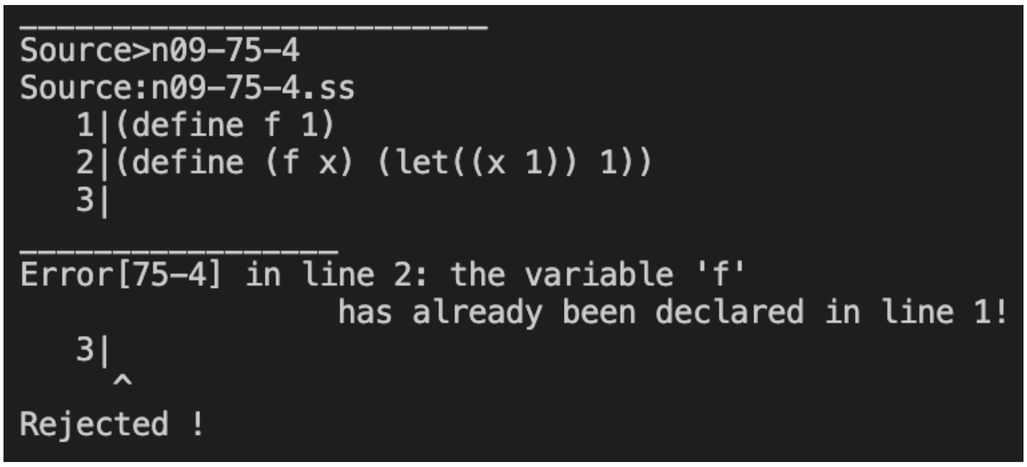
```
_________________________
Source>n09-75-1
Source:n09-75-1.ss
   1|(define (abs x) (let((x 1)) 1))
   2|
_________________
Error[75-1] in line 1: the built-in procedure 'abs'
                has already been defined!
   2|
     ^
Rejected !
```



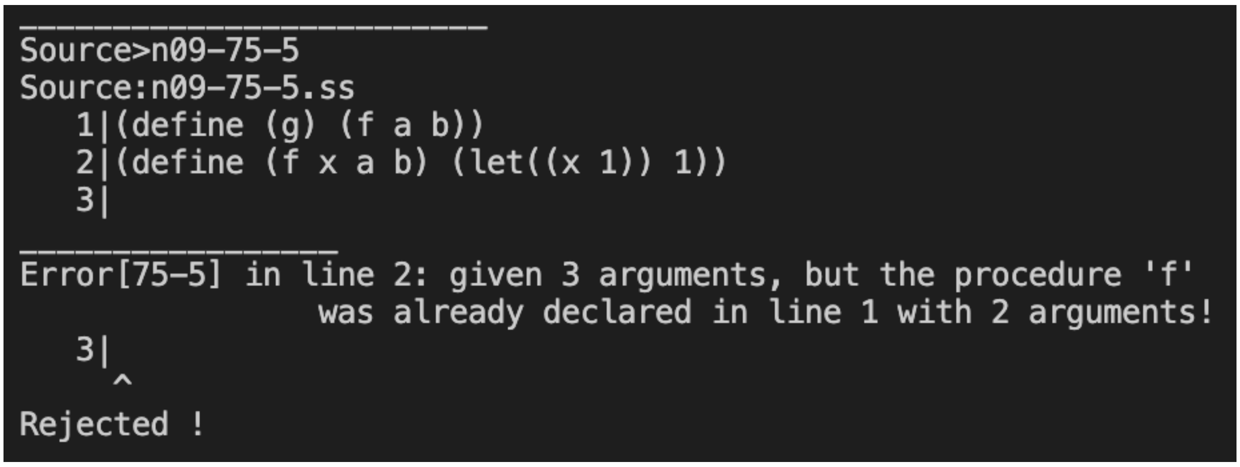
```
_________________________
Source>n09-75-2
Source:n09-75-2.ss
   1|(define (f) 1)
   2|(define (f x) (let((x 1)) 1))
   3|
_________________
Error[75-2] in line 2: the procedure 'f' declared in line 1
                 has already been defined!
   3|
     ^
Rejected !
```



```
_________________________
Source>n09-75-3
Source:n09-75-3.ss
   1|(define (e x) (let((x 1)) 1))
   2|
_________________
Error[75-3] in line 1: the built-in variable 'e'
                has already been defined!
   2|
     ^
Rejected !
```

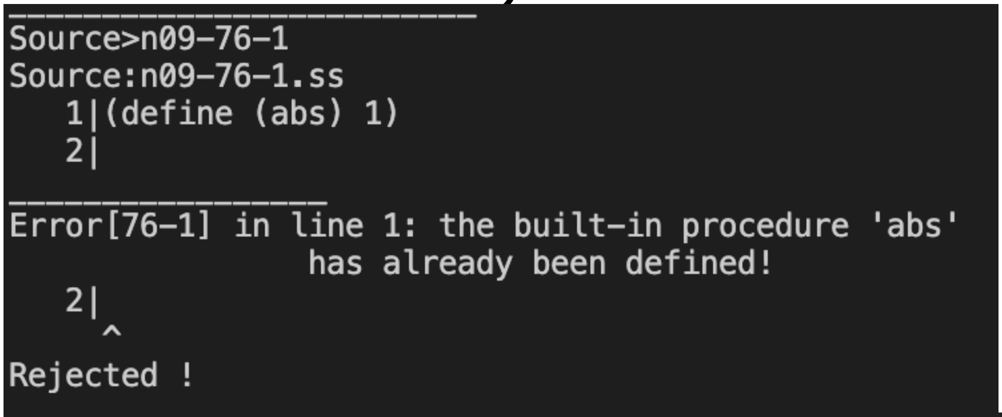


```
_________________________
Source>n09-75-4
Source:n09-75-4.ss
   1|(define f 1)
   2|(define (f x) (let((x 1)) 1))
   3|
_________________
Error[75-4] in line 2: the variable 'f'
                 has already been declared in line 1!
   3|
     ^
Rejected !
```

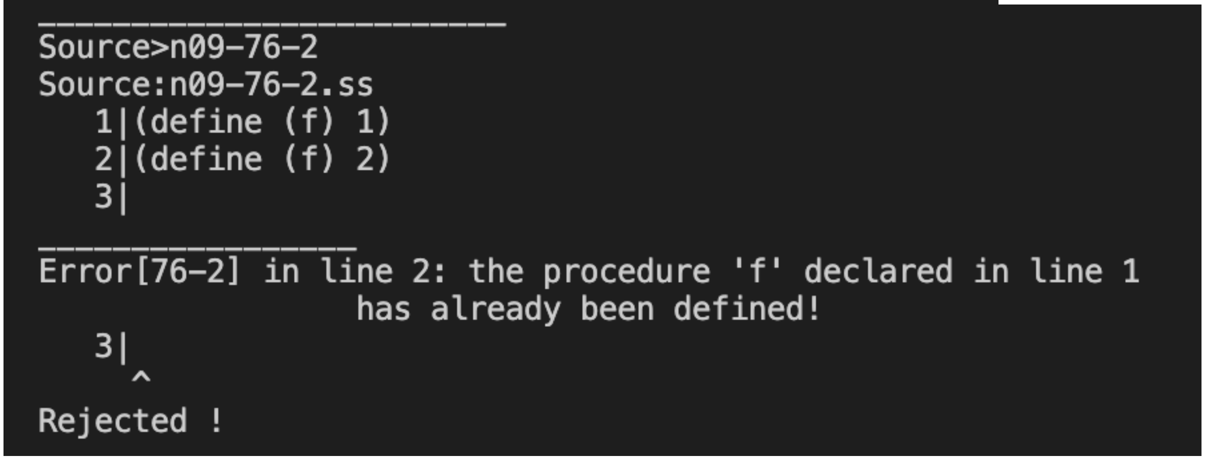


```
_________________________
Source>n09-75-5
Source:n09-75-5.ss
   1|(define (g) (f a b))
   2|(define (f x a b) (let((x 1)) 1))
   3|
_________________
Error[75-5] in line 2: given 3 arguments, but the procedure 'f'
                was already declared in line 1 with 2 arguments!
   3|
     ^
Rejected !
```

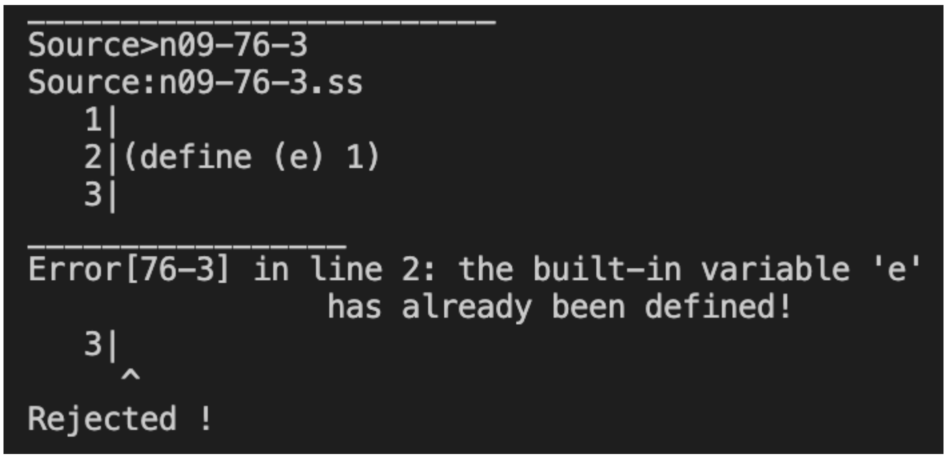
P76: PROC -> HPROC E)



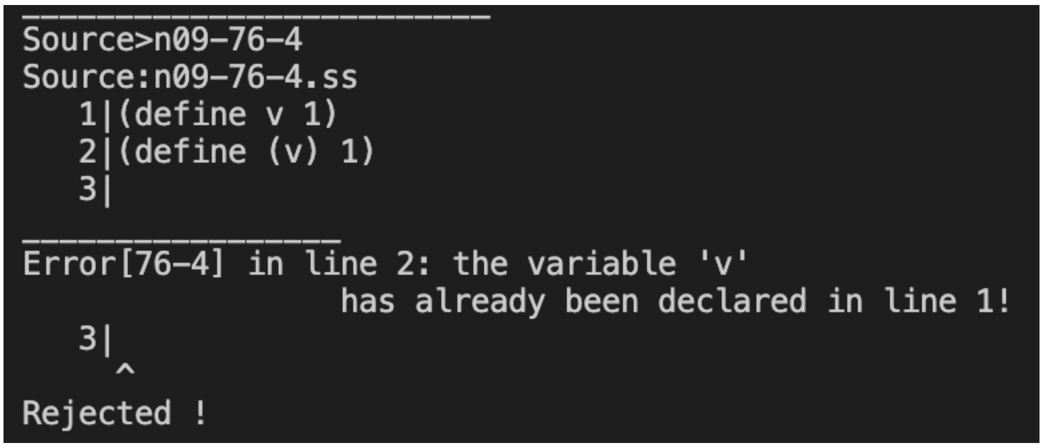
```
_________________________
Source>n09-76-1
Source:n09-76-1.ss
   1|(define (abs) 1)
   2|
_________________
Error[76-1] in line 1: the built-in procedure 'abs'
                has already been defined!
   2|
     ^
Rejected !
```



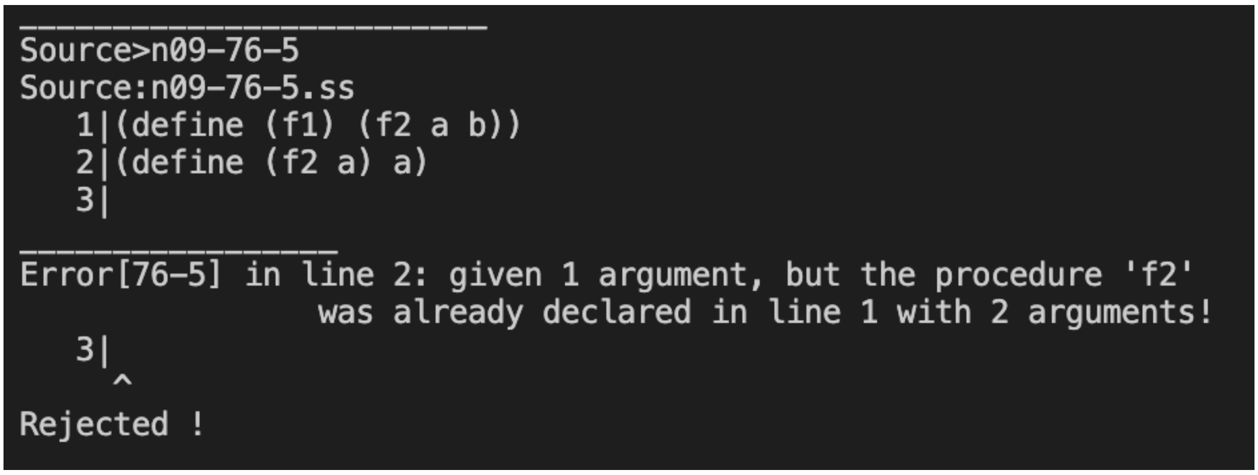
```
_________________________
Source>n09-76-2
Source:n09-76-2.ss
   1|(define (f) 1)
   2|(define (f) 2)
   3|
_________________
Error[76-2] in line 2: the procedure 'f' declared in line 1
                 has already been defined!
   3|
     ^
Rejected !
```



```
 -------------------------
 Source>n09-76-3
 Source:n09-76-3.ss
    1|
    2|(define (e) 1)
    3|
 _________________
 Error[76-3] in line 2: the built-in variable 'e'
                 has already been defined!
    3|
      ^
 Rejected !
```

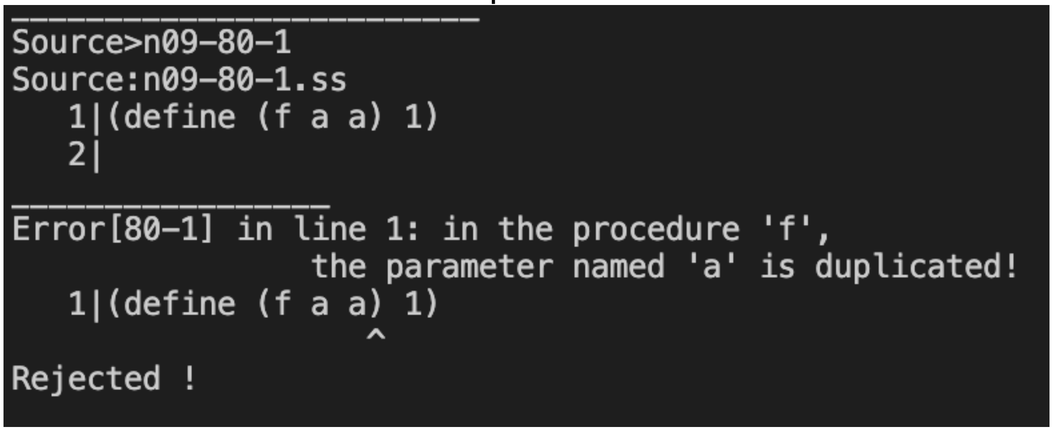


```
 -------------------------
 Source>n09-76-4
 Source:n09-76-4.ss
    1|(define v 1)
    2|(define (v) 1)
    3|
 _________________
 Error[76-4] in line 2: the variable 'v'
                 has already been declared in line 1!
    3|
      ^
 Rejected !
```

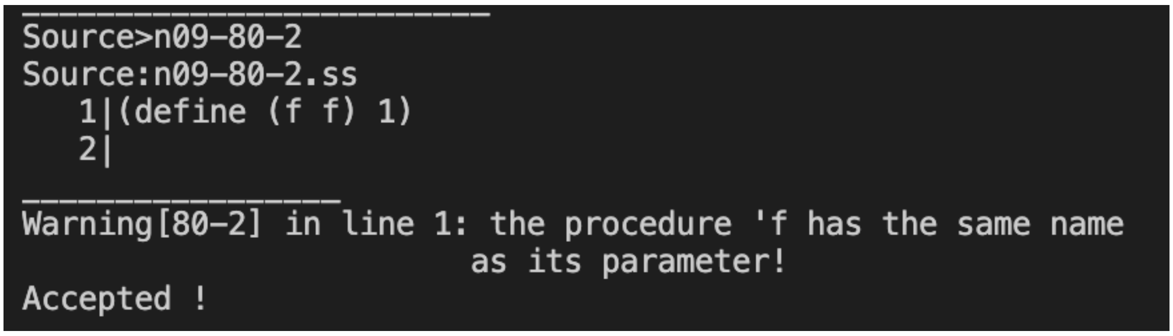


```
 -------------------------
 Source>n09-76-5
 Source:n09-76-5.ss
    1|(define (f1) (f2 a b))
    2|(define (f2 a) a)
    3|
 _________________
 Error[76-5] in line 2: given 1 argument, but the procedure 'f2'
                 was already declared in line 1 with 2 arguments!
    3|
      ^
 Rejected !
```

P80: PCPAR -> PCPAR $id

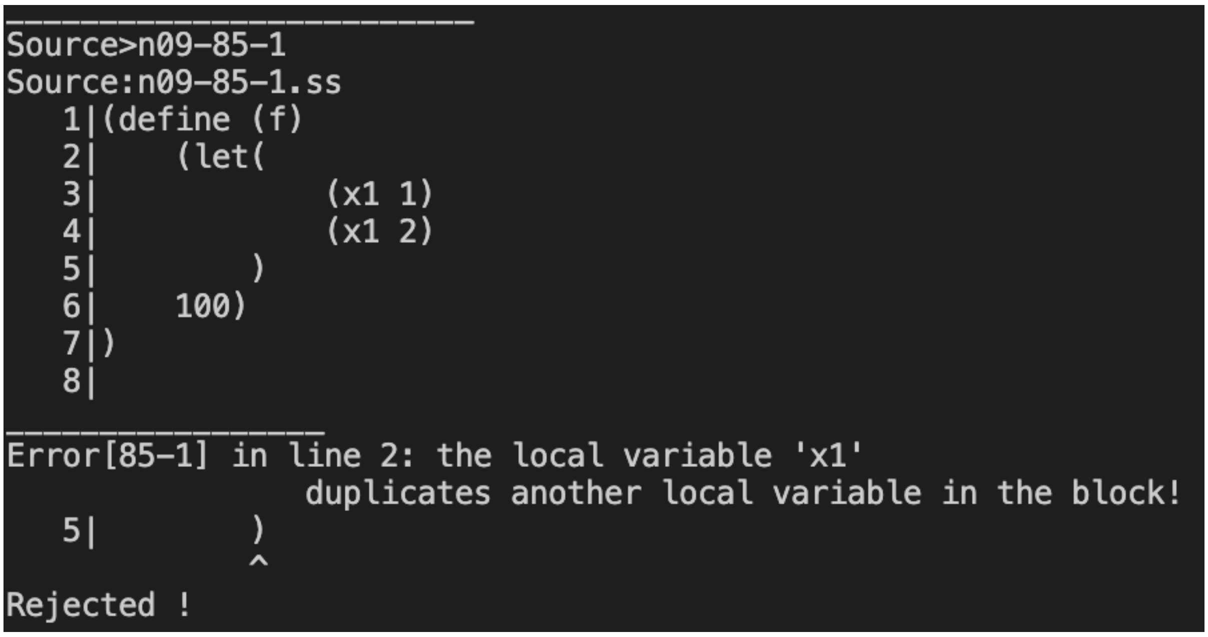


```
-------------------------
Source>n09-80-1
Source:n09-80-1.ss
   1|(define (f a a) 1)
   2|
_________________
Error[80-1] in line 1: in the procedure 'f',
                the parameter named 'a' is duplicated!
   1|(define (f a a) 1)
                   ^
Rejected !
```



```
Source>n09-80-2
Source:n09-80-2.ss
   1|(define (f f) 1)
   2|
_________________
Warning[80-2] in line 1: the procedure 'f has the same name
                        as its parameter!
Accepted !
```

P85: BLVAR -> BLVAR LOCDEF

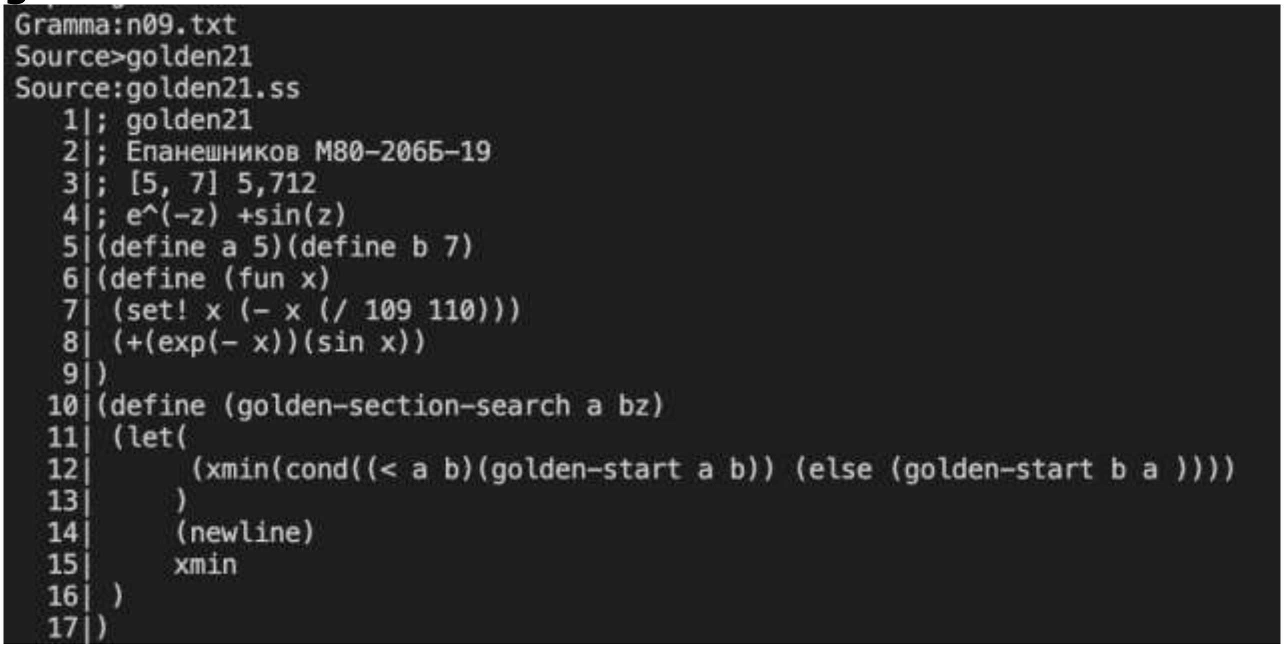


```
Source>n09-85-1
Source:n09-85-1.ss
   1|(define (f)
   2|    (let(
   3|            (x1 1)
   4|            (x1 2)
   5|        )
   6|    100)
   7|)
   8|
_________________
Error[85-1] in line 2: the local variable 'x1'
                duplicates another local variable in the block!
   5|        )
             ^
Rejected !
```

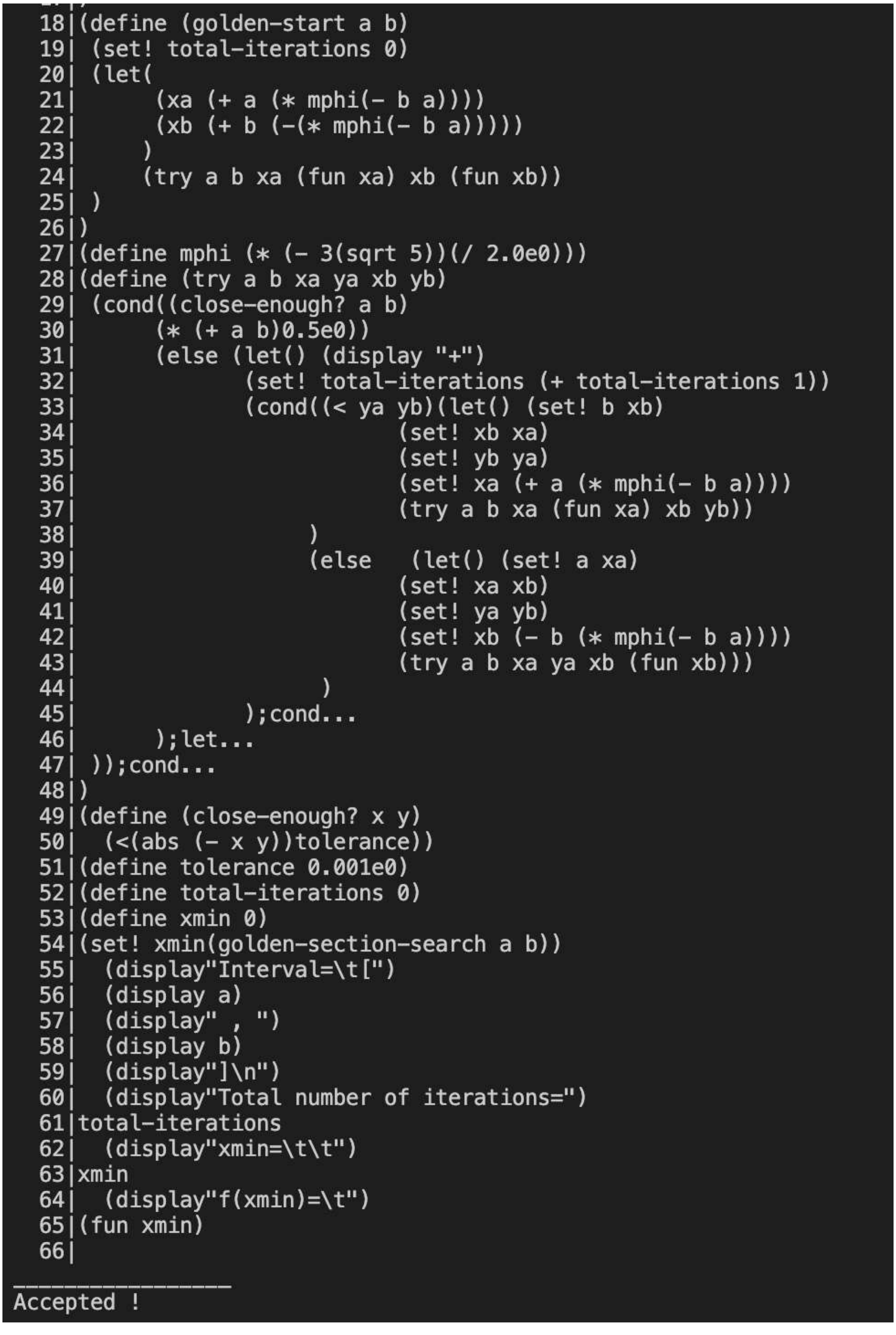
Полные скриншоты анализа своих вариантов программ golden21 и coin21

>

golden21

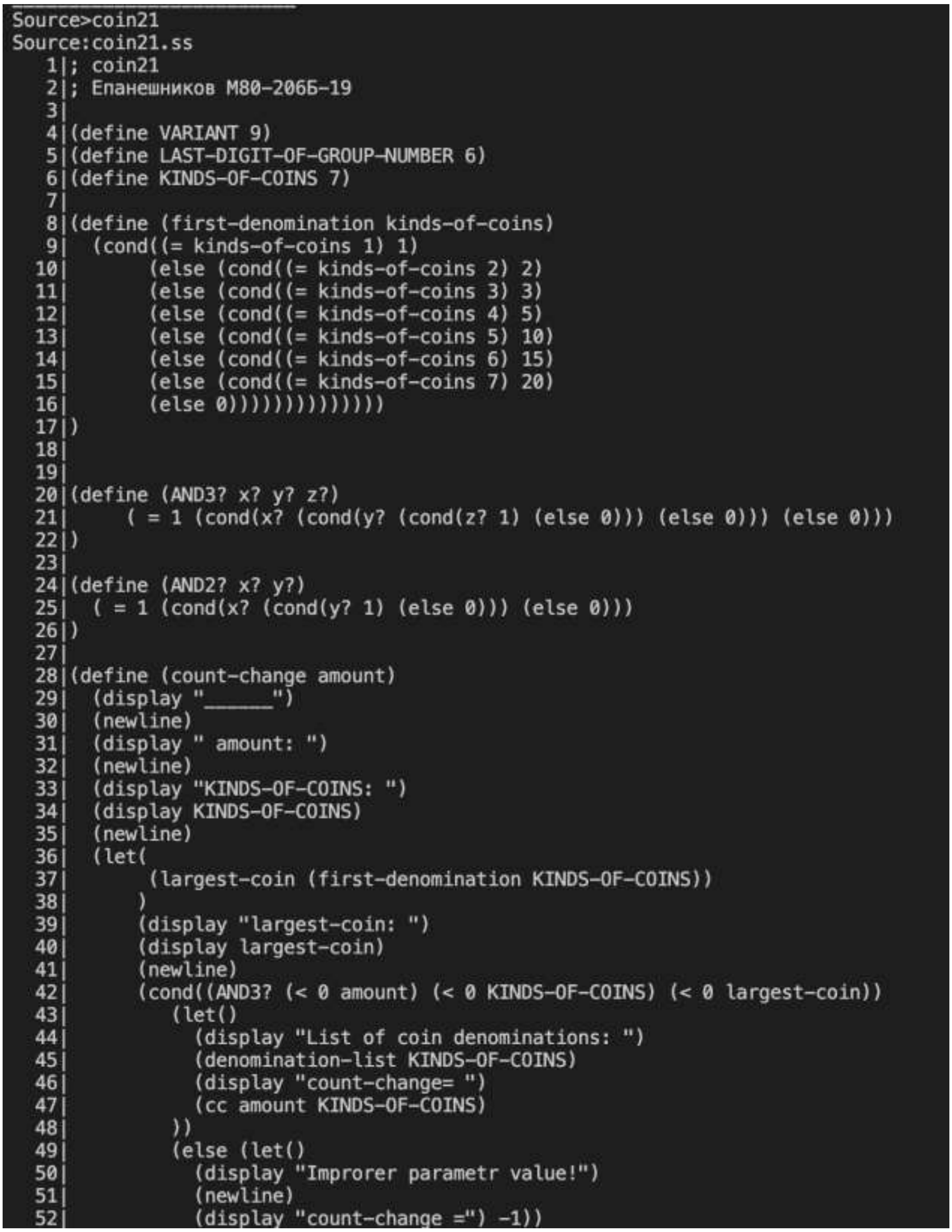


```
Gramma:n09.txt
Source>golden21
Source:golden21.ss
   1|; golden21
   2|; Епанешников М80-206Б-19
   3|; [5, 7] 5,712
   4|; e^(-z) +sin(z)
   5|(define a 5)(define b 7)
   6|(define (fun x)
   7| (set! x (- x (/ 109 110)))
   8| (+(exp(- x))(sin x))
   9|)
  10|(define (golden-section-search a bz)
  11| (let(
  12|      (xmin(cond((< a b)(golden-start a b)) (else (golden-start b a ))))
  13|     )
  14|     (newline)
  15|     xmin
  16| )
  17|)
```

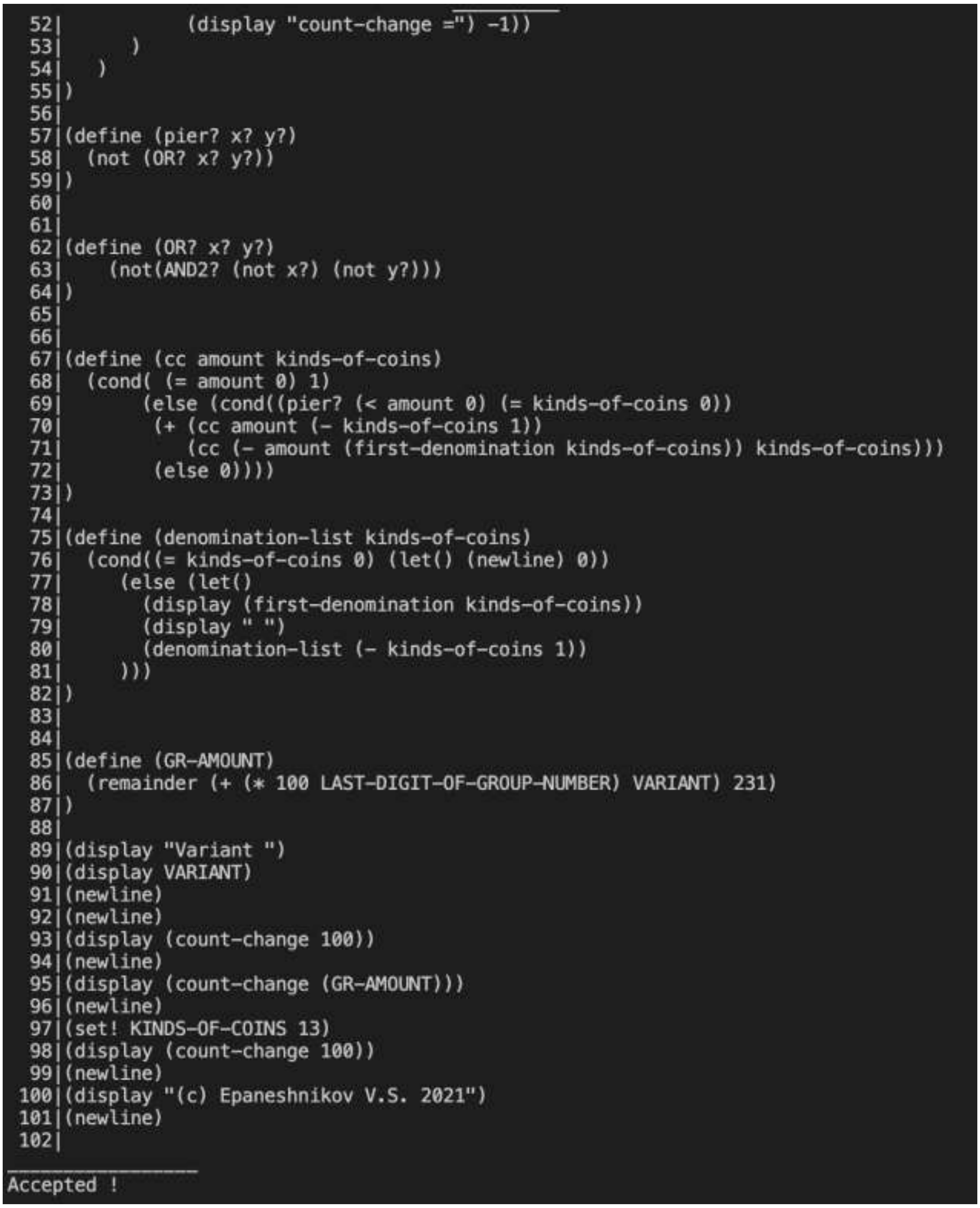


```
(define (golden-start a b)
 (set! total-iterations 0)
 (let(
      (xa (+ a (* mphi(- b a))))
      (xb (+ b (-(* mphi(- b a)))))
     )
     (try a b xa (fun xa) xb (fun xb))
 )
)
(define mphi (* (- 3(sqrt 5))(/ 2.0e0)))
(define (try a b xa ya xb yb)
 (cond((close-enough? a b)
      (* (+ a b)0.5e0))
      (else (let() (display "+")
             (set! total-iterations (+ total-iterations 1))
             (cond((< ya yb)(let() (set! b xb)
                         (set! xb xa)
                         (set! yb ya)
                         (set! xa (+ a (* mphi(- b a))))
                         (try a b xa (fun xa) xb yb))
                  )
                  (else   (let() (set! a xa)
                         (set! xa xb)
                         (set! ya yb)
                         (set! xb (- b (* mphi(- b a))))
                         (try a b xa ya xb (fun xb)))
                   )
             );cond...
      );let...
 ));cond...
)
(define (close-enough? x y)
  (<(abs (- x y))tolerance))
(define tolerance 0.001e0)
(define total-iterations 0)
(define xmin 0)
(set! xmin(golden-section-search a b))
  (display"Interval=\t[")
  (display a)
  (display" , ")
  (display b)
  (display"]\n")
  (display"Total number of iterations=")
total-iterations
  (display"xmin=\t\t")
xmin
  (display"f(xmin)=\t")
(fun xmin)

_________________
Accepted !
```

coin21



```
Source>coin21
Source:coin21.ss
; coin21
; Епанешников М80-206Б-19

(define VARIANT 9)
(define LAST-DIGIT-OF-GROUP-NUMBER 6)
(define KINDS-OF-COINS 7)

(define (first-denomination kinds-of-coins)
  (cond((= kinds-of-coins 1) 1)
       (else (cond((= kinds-of-coins 2) 2)
       (else (cond((= kinds-of-coins 3) 3)
       (else (cond((= kinds-of-coins 4) 5)
       (else (cond((= kinds-of-coins 5) 10)
       (else (cond((= kinds-of-coins 6) 15)
       (else (cond((= kinds-of-coins 7) 20)
       (else 0)))))))))))))
)


(define (AND3? x? y? z?)
     ( = 1 (cond(x? (cond(y? (cond(z? 1) (else 0))) (else 0))) (else 0)))
)

(define (AND2? x? y?)
  ( = 1 (cond(x? (cond(y? 1) (else 0))) (else 0)))
)

(define (count-change amount)
  (display "______")
  (newline)
  (display " amount: ")
  (newline)
  (display "KINDS-OF-COINS: ")
  (display KINDS-OF-COINS)
  (newline)
  (let(
       (largest-coin (first-denomination KINDS-OF-COINS))
      )
      (display "largest-coin: ")
      (display largest-coin)
      (newline)
      (cond((AND3? (< 0 amount) (< 0 KINDS-OF-COINS) (< 0 largest-coin))
         (let()
           (display "List of coin denominations: ")
           (denomination-list KINDS-OF-COINS)
           (display "count-change= ")
           (cc amount KINDS-OF-COINS)
         ))
         (else (let()
           (display "Improrer parametr value!")
           (newline)
           (display "count-change =") -1))
```



```
           (display "count-change =") -1))
      )
   )
)

(define (pier? x? y?)
  (not (OR? x? y?))
)


(define (OR? x? y?)
    (not(AND2? (not x?) (not y?)))
)


(define (cc amount kinds-of-coins)
  (cond( (= amount 0) 1)
       (else (cond((pier? (< amount 0) (= kinds-of-coins 0))
        (+ (cc amount (- kinds-of-coins 1))
           (cc (- amount (first-denomination kinds-of-coins)) kinds-of-coins)))
        (else 0))))
)

(define (denomination-list kinds-of-coins)
  (cond((= kinds-of-coins 0) (let() (newline) 0))
     (else (let()
       (display (first-denomination kinds-of-coins))
       (display " ")
       (denomination-list (- kinds-of-coins 1))
     )))
)


(define (GR-AMOUNT)
  (remainder (+ (* 100 LAST-DIGIT-OF-GROUP-NUMBER) VARIANT) 231)
)

(display "Variant ")
(display VARIANT)
(newline)
(newline)
(display (count-change 100))
(newline)
(display (count-change (GR-AMOUNT)))
(newline)
(set! KINDS-OF-COINS 13)
(display (count-change 100))
(newline)
(display "(c) Epaneshnikov V.S. 2021")
(newline)

_________________
Accepted !
```

Распечатка файла semantics.cpp.
>
/* $n09 */
#include "semantics.h"
using namespace std;

```
// функции для перевода types в строки вида "bool real
real bool..."
std::string to_binary_string(int n) {
   std::string buffer; // символы ответа в обратном
порядке
   // выделим память заранее по максимуму
   buffer.reserve(std::numeric_limits<unsigned
int>::digits);
   do
   {
      buffer += char('0' + n % 2); // добавляем в конец
      n = n / 2;
   } while (n > 0);
   return std::string(buffer.crbegin(), buffer.crend()); //
разворачиваем результат
}

std::string types_to_string(int types, int count) {
   string result;
   string str = to_binary_string(types);
   int size = str.size();
   int diff = count - str.size();
   for (int i = size - 1; i >= 0; --i) {
      if (str[i] == '0') {
         result += "real ";
      }
      else if (str[i] == '1') {
         result += "bool ";
      }
   }
   while (diff != 0) {
      result += "real ";
      --diff;
   }
   result.pop_back();
   return result;
}

void tSM::init() {
   globals.clear();
   locals.clear();
   params.clear();
   scope = 0;

   globals["abs"] = tgName(PROC | DEFINED | BUILT, "",
1);
   globals["atan"] = tgName(PROC | DEFINED | BUILT, "",
1);
   globals["cos"] = tgName(PROC | DEFINED | BUILT, "",
1);
   globals["exp"] = tgName(PROC | DEFINED | BUILT, "",
1);
   globals["expt"] = tgName(PROC | DEFINED | BUILT, "",
2);
   globals["log"] = tgName(PROC | DEFINED | BUILT, "", 1);
   globals["remainder"] = tgName(PROC | DEFINED |
BUILT, "", 2);
   globals["quotient"] = tgName(PROC | DEFINED | BUILT,
"", 2);
   globals["sin"] = tgName(PROC | DEFINED | BUILT, "", 1);
   globals["sqrt"] = tgName(PROC | DEFINED | BUILT, "",
1);
   globals["tan"] = tgName(PROC | DEFINED | BUILT, "", 1);
   globals["display"] = tgName(PROC | DEFINED | BUILT,
"", 1);
   globals["newline"] = tgName(PROC | DEFINED | BUILT,
"", 0);
   globals["e"] = tgName(VAR | DEFINED | BUILT, "");
   globals["pi"] = tgName(VAR | DEFINED | BUILT, "");
   return;
}

int tSM::p01() { //       S -> PROG
   bool error=false;
   for(tGlobal::iterator it=globals.begin(); it!=globals.end();
++it) {
      if(it->second.test(USED) && !it-
>second.test(DEFINED)) {
         if(it->second.test(VAR)) {
            ferror_message += "Error[01-1] in line " + it-
>second.line +
             ": the variable '" + it->first +
            "' is used, \n\t\t\tbut not defined!\n";
            // переменная 'v' используется, но не
определена!
            // the variable 'v' is used, but not defined!
            error = true;
         }
         else if(it->second.test(PROC)) {
            ferror_message += "Error[01-2] in line " + it-
>second.line +
            ": the procedure '" + it->first +
            "' is used, \n\t\t\tbut not defined!\n";
            // процедура 'f' используется, но не
определена!
            // the procedure 'f' is used, but not defined!
            error = true;
         }
      }
   }

   if(error) return 1;
   return 0;
}

int tSM::p02() { //     PROG -> CALCS
   return 0;}
int tSM::p03() { //     PROG -> DEFS
   return 0;}
int tSM::p04() { //     PROG -> DEFS CALCS
   return 0;}

int tSM::p05() { //       E -> $id
   string name = S1->name;
   switch (scope) {
   case 2:
      if (locals.count(name)) break;

   case 1:
      if (params.count(name)) break;

   default:
      tgName& ref = globals[name];
      if (ref.empty()) {
         ref = tgName(VAR|USED, S1->line);
         break;
      }
      if (ref.test(VAR)) {
         ref.set(USED);
         break;
      }
      if (ref.test(BUILT)) {
         ferror_message += "Error[05-1] in line "+ S1->line
+": the built-in '" + name +
         "' procedure \n\t\t\tcannot be used as a
variable!\n";
         // встроенную процедуру 'abs' нельзя
использовать в качестве переменной!
         // the built-in 'abs' procedure cannot be used as a
variable!
         return 1;
      }

      ferror_message += "Error[05-2] in line "+ S1->line +":
the name '" + name +
      "' cannot be used to refer to a variable,\n\t\t\t" +
      "it was previously declared as a procedure in line "+
ref.line + "!\n";
      // имя 'f' нельзя использовать для ссылки на
переменную, в строке 1 оно ранее объявлено как
процедура!
      // the name 'f' cannot be used to refer to a variable, it
was previously declared as a procedure in line 1!
      return 1;
   }

   return 0;
}

int tSM::p06() { //       E -> $int
   return 0;}
int tSM::p07() { //       E -> $dec
   return 0;}
int tSM::p08() { //       E -> AREX
   return 0;}
int tSM::p09() { //       E -> COND
   return 0;}
```

```
int tSM::p10() { //       E -> EASYLET
   return 0;}
int tSM::p11() { //       E -> CPROC
   return 0;}
int tSM::p12() { //    AREX -> HAREX E )
   return 0;}
int tSM::p13() { //   HAREX -> ( AROP
   return 0;}
int tSM::p14() { //   HAREX -> HAREX E
   return 0;}
int tSM::p15() { //    AROP -> +
   return 0;}
int tSM::p16() { //    AROP -> -
   return 0;}
int tSM::p17() { //    AROP -> *
   return 0;}
int tSM::p18() { //    AROP -> /
   return 0;}
int tSM::p19() { // EASYLET -> HEASYL E )
   return 0;}
int tSM::p20() { //  HEASYL -> ( let ( )
   return 0;}
int tSM::p21() { //  HEASYL -> HEASYL INTER
   return 0;}

int tSM::p22() { //   CPROC -> HCPROC )
   switch (scope) {
   case 2:
      if (locals.count(S1->name)) {
         if (globals[S1->name].test(BUILT)) {
            ferror_message += "Error[22-1] in line "+ S1-
>line +
            ": the local variable '" + S1->name + "' masks the
built-in procedure!\n";
            // локальная переменная 'a' маскирует
встроенную процедуру!
            // the local variable 'a' masks the built-in
procedure!
            return 1;
         }
         ferror_message += "Error[22-2] in line "+ S1->line
+
         ": the local variable '" + S1->name + "' masks the
procedure \n\t\t";
         if (globals[S1->name].test(DEFINED) || globals[S1-
>name].test(USED)) {
            ferror_message += "declared in line "
            + globals[S1->name].line + " ";
         }
         ferror_message += "with the same name!\n";
         // локальная переменная 'a' маскирует процедуру
(объявленную в строке 1) с таким же именем!
         // the local variable 'a' masks the procedure
(declared in line 1) with the same name!
         return 1;
      }
   case 1:
      if (params.count(S1->name)) {
         if (globals[S1->name].test(BUILT)) {
            ferror_message += "Error[22-3] in line "+ S1-
>line +
            ": the parameter '" + S1->name + "' masks the
built-in procedure!\n";
            // параметр 'a' маскирует встроенную
процедуру!
            // the parameter 'a' masks the built-in procedure!
            return 1;
         }
         ferror_message += "Error[22-4] in line "+ S1->line
+
         ": the parameter '" + S1->name + "' masks the
procedure \n\t\t";
         if (globals[S1->name].test(DEFINED) || globals[S1-
>name].test(USED)) {
            ferror_message += "declared in line "
            + globals[S1->name].line + " ";
         }
         ferror_message += "with the same name!\n";
         // параметр 'a' маскирует процедуру
(объявленную в строке 1) с таким же именем!
         // the parameter 'a' masks the procedure (declared
in line 1) with the same name!
```

```
         return 1;
      }
   default:
      if (globals[S1->name].empty()) {
         globals[S1->name] = tgName(PROC | USED, S1-
>line, S1->count);
         return 0;
      }

      if (!globals[S1->name].test(PROC)) {
         if (!globals[S1->name].test(BUILT)) {
            ferror_message = "Error[22-5] in line "+
            S1->line + ": the name '" + S1->name +
            "' is not a procedure \n\t\t'" + S1->name +
            "' was declared in line " + globals[S1->name].line
+
            " as a variable!\n";
            // имя 'a' не процедура, 'a' было объявлено в
строке 1 как переменная!
            // the name 'a' is not a procedure, 'a' was
declared in line 1 as a variable!
            return 1;
         }
         ferror_message = "Error[22-6] in line "+
         S1->line + ": the name '" + S1->name +
         "' is not a procedure \n\t\t'" + S1->name +
         "' is a built-in variable!\n";
         // имя 'name' не процедура, 'name' это встроенная
переменная!
         // the name 'name' is not 'a' procedure, 'name' is a
built-in variable!
         return 1;
      }

      if (globals[S1->name].arity != S1->count) {
         if (globals[S1->name].test(BUILT)) {
            ferror_message = "Error[22-7] in line "+ S1->line
+
            ": the built-in procedure '" + S1->name + "' " +
            "\n\t\t\texpects " + std::to_string(globals[S1-
>name].arity) +
            " argument" + (globals[S1->name].arity != 1 ? "s"
: "") +
             ", writen " + std::to_string(S1->count) + "!\n";
            // встроенная процедура 'abs' ожидает n
аргумента(ов), записано m!
            // the built-in procedure 'abs' expects n
argument(s), writen m!
            return 1;
         }
         ferror_message = "Error[22-8] in line "+ S1->line +
         ": given " + std::to_string(S1->count) + " argument"
+
         (S1->count != 1 ? "s" : "") + ", but the procedure '"
+
         S1->name + "' \n\t\twas already declared in line "
+
         globals[S1->name].line + " with " +
std::to_string(globals[S1->name].arity) + " argument" +
         (globals[S1->name].arity != 1 ? "s" : "") + "!\n";
         // дано m аргумент(ов), но процедура 'f' была уже
объявлена в строке 1 с n параметром(ами)!
         // given 5 arguments, but the procedure 'f' was
already declared in line 1 with n agrument(s)!
         return 1;
      }

      globals[S1->name].set(USED);
      return 0;
   }
   return 0;
}
int tSM::p23() { //   HCPROC -> ( $id
   S1->name = S2->name;
   S1->count = 0;
   return 0;
}

int tSM::p24() { //   HCPROC -> HCPROC E

   ++S1->count;
   return 0;
}
```

```
int tSM::p25() { //     COND -> ( cond BRANCHES )
   return 0;}
int tSM::p26() { // BRANCHES -> CLAUS ELSE
   return 0;}
int tSM::p27() { //    CLAUS -> ( BOOL E )
   return 0;}
int tSM::p28() { //     ELSE -> ( else E )
   return 0;}
int tSM::p29() { //      STR -> $str
   return 0;}
int tSM::p30() { //      STR -> SCOND
   return 0;}
int tSM::p31() { //    SCOND -> ( cond SBRANCHES )
   return 0;}
int tSM::p32() { //SBRANCHES -> SELSE
   return 0;}
int tSM::p33() { //SBRANCHES -> SCLAUS SBRANCHES
   return 0;}
int tSM::p34() { //   SCLAUS -> ( BOOL STR )
   return 0;}
int tSM::p35() { //    SELSE -> ( else STR )
   return 0;}
int tSM::p36() { //     BOOL -> $bool
   return 0;}

int tSM::p37() { //     BOOL -> $idq
   string name = S1->name;

   switch (scope) {
   case 2:
      if (locals.count(name)) break;

   case 1:
      if (params.count(name)) break;

   default:
      tgName& ref = globals[name];

      if (ref.empty()) {
         ref = tgName(PROC|USED, S1->line);
         break;
      }

      if (globals[S1->name].arity != S1->count) {
         ferror_message = "Error[37-1] in line "+ S1->line +
         ": given " + std::to_string(S1->count) + " argument"
+
         (S1->count != 1 ? "s" : "") + ", but the predicate '" +
         S1->name + "' \n\t\twas already declared in line "
+
         globals[S1->name].line + " with " +
std::to_string(globals[S1->name].arity) + " argument" +
         (globals[S1->name].arity != 1 ? "s" : "") + "!\n";
         // дано m аргумент(ов), но предикат 'f' был уже
объявлена в строке 1 с n параметром(ами)!
         // given 5 arguments, but the predicate 'f' was
already declared in line 1 with n agrument(s)!
         return 1;
      }
   }

   return 0;
}

int tSM::p38() { //     BOOL -> REL
   return 0;}
int tSM::p39() { //     BOOL -> ( not BOOL )
   return 0;}
int tSM::p40() { //     BOOL -> CPRED
   return 0;}

int tSM::p41() { //    CPRED -> HCPRED )
   switch (scope) {
   case 1:
      if (params.count(S1->name)) {
         ferror_message += "Error[41-1] in line "+ S1->line
+
         ": the parameter '" + S1->name + "' masks the
predicate \n\t\t";
         if (globals[S1->name].test(DEFINED) || globals[S1-
>name].test(USED)) {
            ferror_message += "declared in line "
            + globals[S1->name].line + " ";
         }
         ferror_message += "with the same name!\n";
         // параметр 'a' маскирует предикат (объявленный
в строке 1) с таким же именем!
         // the parameter 'a' masks the predicate (declared
in line 1) with the same name!
         return 1;
      }
   default:
      if (globals[S1->name].empty()) {
         globals[S1->name] = tgName(PROC | USED, S1-
>line, S1->count, S1->types);
         return 0;
      }

      if (globals[S1->name].arity != S1->count) {
         ferror_message = "Error[41-2] in line "+ S1->line +
         ": given " + std::to_string(S1->count) + " argument"
+
         (S1->count != 1 ? "s" : "") + ", but the predicate '" +
         S1->name + "' \n\t\twas already declared in line "
+
         globals[S1->name].line + " with " +
std::to_string(globals[S1->name].arity) + " argument" +
         (globals[S1->name].arity != 1 ? "s" : "") + "!\n";
         // дано m аргумент(ов), но предикат 'f' был уже
объявлен в строке 1 с n параметром(ами)!
         // given 5 arguments, but the predicate 'f' was
already declared in line 1 with n agrument(s)!
         return 1;
      }

      if (globals[S1->name].types != S1->types) {
         ferror_message = "Error[41-3] in line " + S1->line +
         ": invalid parameter types in the predicate '" +
         S1->name + "' \ndeclared in line " + globals[S1-
>name].line + ", " +
         "need [" + types_to_string(globals[S1-
>name].types, globals[S1->name].arity) +
         "], writen [" + types_to_string(S1->types, S1-
>count) + "]!\n";
```

```
          // неверные типы параметров в предикате 'p',
объявленном в строке 1,
          //     нужно [bool, real..], записано [real, bool...]!
          // invalid parameter types in the predicate 'p'
declared in line 1,
          //     need [bool, real..], written by [real, bool...]!
          return 1;
       }

   }
   globals[S1->name].set(USED);
   return 0;
}

int tSM::p42() { //   HCPRED -> ( $idq

   S1->name = S2->name;
   S1->count = 0;
   S1->types = 0;
   return 0;
}

int tSM::p43() { //   HCPRED -> HCPRED ARG
   S1->types = S1->types | (S2->types << S1->count);
   ++S1->count;
   return 0;
}

int tSM::p44() { //     ARG -> E
   S1->types = 0;
   return 0;
}
int tSM::p45() { //     ARG -> BOOL

   S1->types = 1;
   return 0;
}

int tSM::p46() { //     REL -> ( = E E )
   return 0;}
int tSM::p47() { //     REL -> ( < E E )
   return 0;}
```

```
int tSM::p48() { //      SET -> HSET E )
   return 0;}

int tSM::p49() { //     HSET -> ( set! $id
   switch (scope) {
   case 2:
      if (locals.count(S3->name)) {
         return 0;
      }
   case 1:
      if (params.count(S3->name)) {
         return 0;
      }
   default:
      if (globals[S3->name].empty()) {
         globals[S3->name] = tgName(VAR | USED, S3-
>line);
         return 0;
      }

      if (globals[S3->name].test(BUILT) && globals[S3-
>name].test(VAR)) {
         ferror_message = "Error[49-1] in line " + S1->line +
": the built-in variable '"
         + S3->name + "' \n\t\tcannot be redefined with
set!\n";
         // встроенную переменную 'f' нельзя
переопределить с помощью set!
         // the built-in variable 'f' cannot be redefined with
set!
         return 1;
      }

      if (globals[S3->name].test(PROC)) {
         ferror_message = "Error[49-2] in line " + S1->line +
": the procedure '" + S3->name +
         "' declared \n\t\t in line " + globals[S3->name].line
+ " cannot be redefined with set!\n";
         // процедура 'f' объявленная в строке 1 не может
быть переопределена с помощью set!
         // the procedure 'f' declared in line 1 cannot be
redefined with set!
```

```
        return 1;
      }
      globals[S3->name].set(USED);
      return 0;
    }
    return 0;
}

int tSM::p50() { //  DISPSET -> ( display E )
    return 0;}
int tSM::p51() { //  DISPSET -> ( display BOOL )
    return 0;}
int tSM::p52() { //  DISPSET -> ( display STR )
    return 0;}
int tSM::p53() { //  DISPSET -> ( newline )
    return 0;}
int tSM::p54() { //  DISPSET -> SET
    return 0;}
int tSM::p55() { //    INTER -> DISPSET
    return 0;}
int tSM::p56() { //    INTER -> E
    return 0;}
int tSM::p57() { //    CALCS -> CALC
    return 0;}
int tSM::p58() { //    CALCS -> CALCS CALC
    return 0;}
int tSM::p59() { //     CALC -> E
    return 0;}
int tSM::p60() { //     CALC -> BOOL
    return 0;}
int tSM::p61() { //     CALC -> STR
    return 0;}
int tSM::p62() { //     CALC -> DISPSET
    return 0;}
int tSM::p63() { //     DEFS -> DEF
    return 0;}
int tSM::p64() { //     DEFS -> DEFS DEF
    return 0;}
int tSM::p65() { //      DEF -> PRED
    return 0;}
int tSM::p66() { //      DEF -> VAR
    return 0;}
```

```
int tSM::p67() { //      DEF -> PROC
   return 0;}

int tSM::p68() { //     PRED -> HPRED BOOL )
   if (globals[S1->name].empty()) {
      params.clear();
      scope = 0;
      globals[S1->name] = tgName(PROC | DEFINED, S1-
>line, S1->count, S1->types);
      return 0;
   }

   if (globals[S1->name].test(PROC) && globals[S1-
>name].test(DEFINED)) {
      ferror_message = "Error[68-1] in line " + S1->line +
      ": the predicate '" + S1->name + "' declared in line " +
      globals[S1->name].line + "\n\t\t has already been
defined!\n";
      // предикат 'a' объявленный на строке 1 уже был
определён!
      // the predicate 'a' declared in line 1 has already been
defined!
      return 1;
   }

   if (globals[S1->name].arity != S1->count) {
      ferror_message = "Error[68-2] in line "+ S1->line +
      ": given " + std::to_string(S1->count) + " argument" +
      (S1->count != 1 ? "s" : "") + ", but the predicate '" +
      S1->name + "' \n\t\twas already declared in line " +
      globals[S1->name].line + " with " +
std::to_string(globals[S1->name].arity) + " argument" +
      (globals[S1->name].arity != 1 ? "s" : "") + "!\n";
      // дано m аргумент(ов), но предикат 'f' был уже
объявлен в строке 1 с n параметром(ами)!
      // given 5 arguments, but the predicate 'f' was already
declared in line 1 with n agrument(s)!
      return 1;
   }

   if (globals[S1->name].types != S1->types) {
      ferror_message = "Error[68-3] in line " + S1->line +
      ": invalid parameter types in the predicate '" +
       S1->name + "' \ndeclared in line " + globals[S1-
>name].line + ", " +
      "need [" + types_to_string(globals[S1->name].types,
globals[S1->name].arity) +
      "], writen [" + types_to_string(S1->types, S1->count)
+ "]\n";
      // неверные типы параметров в предикате 'p',
объявленном в строке 1,
      // нужно [bool, real..], записано [real, bool...]
      // invalid parameter types in the predicate 'p' declared
in line 1,
      // need [bool, real..], written by [real, bool...]
      return 1;
   }

   params.clear();
   scope = 0;
   globals[S1->name].set(DEFINED);
   return 0;
}

int tSM::p69() { //    HPRED -> PDPAR )
   scope = 1;
   return 0;
}
int tSM::p70() { //    PDPAR -> ( define ( $idq
   S1->name = S4->name;
   S1->count = 0;
   S1->types = 0;
   return 0;
}
int tSM::p71() { //    PDPAR -> PDPAR $idq
   if (params.count(S2->name)) {
      ferror_message = "Error[71-1] in line " + S1->line + ":
in the predicate '" + S1->name +
      "', \n\t\tthe parameter named '" + S2->name + "' is
duplicated!\n";
      // в предикате 'a' дублируется параметр с именем
'b'!
      // in the predicate 'a', the parameter named 'b' is
duplicated!
```

```
      return 1;
   }

   if (S2->name == S1->name) {
      ferror_message +=
      "Warning[71-2] in line " + S2->line + ": the predicate
'"
      + S1->name + "' has the same name \n\t\t\tas its
parameter!\n";
      // предикат 'f' имеет такое же имя как параметр!
      // the predicate 'f' has the same name as its
parameter!
   }

   S1->types = S1->types | (1<<S1->count);
   ++S1->count;
   params.insert(S2->name);
   return 0;
}

int tSM::p72() { //    PDPAR -> PDPAR $id
   if (params.count(S2->name)) {
      ferror_message = "Error[72-1] in line " + S1->line + ":
in the predicate '" + S1->name +
      "', \n\t\tthe parameter named '" + S2->name + "' is
duplicated!\n";
      // в предикате  'a' дублируется параметр с именем
'b'!
      // in the variable 'a', the parameter named 'b' is
duplicated!
      return 1;
   }

   ++S1->count;
   params.insert(S2->name);
   return 0;
}

int tSM::p73() { //      VAR -> VARDCL E )
   return 0;}

int tSM::p74() { //   VARDCL -> ( define $id
   if (globals[S3->name].empty()) {
      globals[S3->name] = tgName(VAR | DEFINED, S3-
>line);
      return 0;
   }

   if (globals[S3->name].test(PROC)) {
      if (globals[S3->name].test(BUILT)) {
         ferror_message = "Error[74-1] in line " + S3->line +
         ": the built-in procedure '" + S3->name + "'
\n\t\thas already been defined!\n";
         // встроенная процедура 'a' уже была
определена!
         // the built-in procedure 'a' has already been
defined!
         return 1;
      }
      ferror_message = "Error[74-2] in line " + S3->line +
      ": the procedure with the same name '" + S3->name +
      "'\n\t\t has already been declared in line " +
globals[S3->name].line + "!\n";
      // процедура с таким же именем 'a' уже была
объявлена в строке 1!
      // the procedure with the same name 'a' has already
been declared in line 1!
      return 1;
   }
   else if (globals[S3->name].test(VAR)) {
      if (globals[S3->name].test(BUILT)) {
         ferror_message = "Error[74-3] in line " + S3->line +
         ": the built-in variable '" + S3->name + "' \n\t\thas
already been defined!\n";
         // встроенная переменная 'a' уже была
определена!
         // the built-in variable 'a' has already been defined!
         return 1;
      }
      if (globals[S3->name].test(DEFINED)) {
         ferror_message = "Error[74-4] in line " + S3->line +
         ": the variable with the same name '" + S3->name +
"' \n\t\t declared in line " +
         globals[S3->name].line + " has already been
defined!\n";
         // переменная с таким же именем 'a' объявленная
в строке 1 уже была определена!
         // the variable with the same name 'a' declared in
line 1 has already been defined!
         return 1;
      }
   }

   globals[S3->name].set(DEFINED);
   return 0;
}

int tSM::p75() { //     PROC -> HPROC BLOCK )
   if (globals[S1->name].empty()) {
      params.clear();
      scope = 0;
      globals[S1->name] = tgName(PROC | DEFINED, S1-
>line, S1->count);
      return 0;
   }

   if (globals[S1->name].test(PROC)) {
      if (globals[S1->name].test(BUILT)) {
         ferror_message = "Error[75-1] in line " + S1->line +
         ": the built-in procedure '" + S1->name + "'
\n\t\thas already been defined!\n";
         // встроенная процедура 'a' уже была
определена!
         // the built-in procedure 'a' has already been
defined!
         return 1;
      }
      if (globals[S1->name].test(DEFINED)) {
         ferror_message = "Error[75-2] in line " + S1->line +
         ": the procedure '" + S1->name + "' declared in line
" +
         globals[S1->name].line + "\n\t\t has already been
defined!\n";
         // процедура 'a' объявленная на строке 1 уже
была определёна!
```

```
            // the procedure 'a' declared in line 1 has already
been defined!
            return 1;
        }
    }
    else if (globals[S1->name].test(VAR)) {
        if (globals[S1->name].test(BUILT)) {
            ferror_message = "Error[75-3] in line " + S1->line +
            ": the built-in variable '" + S1->name + "' \n\t\thas
already been defined!\n";
            // встроенная переменная 'a' уже была
определена!
            // the built-in variable 'a' has already been defined!
            return 1;
        }
        ferror_message = "Error[75-4] in line " + S1->line +
        ": the variable '" + S1->name  +
        "'\n\t\t has already been declared in line " +
globals[S1->name].line + "!\n";
        // переменная 'a' уже была объявлена в строке 1!
        // the variable 'a' has already been declared in line 1!
        return 1;

    }

    if (globals[S1->name].arity != S1->count) {
        ferror_message = "Error[75-5] in line "+ S1->line +
        ": given " + std::to_string(S1->count) + " argument" +
        (S1->count != 1 ? "s" : "") + ", but the procedure '" +
        S1->name + "' \n\t\twas already declared in line " +
        globals[S1->name].line + " with " +
std::to_string(globals[S1->name].arity) + " argument" +
        (globals[S1->name].arity != 1 ? "s" : "") + "!\n";
        // дано m аргумент(ов), но процедура 'f' была уже
объявлена в строке 1 с n параметром(ами)!
        // given 5 arguments, but the procedure 'f' was
already declared in line 1 with n agrument(s)!
        return 1;
    }

    params.clear();
    scope = 0;
```

```
   globals[S1->name].set(DEFINED);
   return 0;
}

int tSM::p76() { //     PROC -> HPROC E )
   if (globals[S1->name].empty()) {
      params.clear();
      scope = 0;
      globals[S1->name] = tgName(PROC | DEFINED, S1-
>line, S1->count);
      return 0;
   }

   if (globals[S1->name].test(PROC)) {
      if (globals[S1->name].test(BUILT)) {
         ferror_message = "Error[76-1] in line " + S1->line +
         ": the built-in procedure '" + S1->name + "'
\n\t\thas already been defined!\n";
         // встроенная процедура 'a' уже была
определена!
         // the built-in procedure 'a' has already been
defined!
         return 1;
      }
      if (globals[S1->name].test(DEFINED)) {
         ferror_message = "Error[76-2] in line " + S1->line +
         ": the procedure '" + S1->name + "' declared in line
" +
         globals[S1->name].line + "\n\t\t has already been
defined!\n";
         // процедура 'a' объявленная на строке 1 уже
была определёна!
         // the procedure 'a' declared in line 1 has already
been defined!
         return 1;
      }
   }
   else if (globals[S1->name].test(VAR)) {
      if (globals[S1->name].test(BUILT)) {
         ferror_message = "Error[76-3] in line " + S1->line +
         ": the built-in variable '" + S1->name + "' \n\t\thas
already been defined!\n";
```

```
         // встроенная переменная 'a' уже была
определена!
         // the built-in variable 'a' has already been defined!
         return 1;
      }
      ferror_message = "Error[76-4] in line " + S1->line +
      ": the variable '" + S1->name  +
      "'\n\t\t has already been declared in line " +
globals[S1->name].line + "!\n";
      // переменная 'a' уже была объявлена в строке 1!
      // the variable 'a' has already been declared in line 1!
      return 1;

   }

   if (globals[S1->name].arity != S1->count) {
      ferror_message = "Error[76-5] in line "+ S1->line +
      ": given " + std::to_string(S1->count) + " argument" +
      (S1->count != 1 ? "s" : "") + ", but the procedure '" +
      S1->name + "' \n\t\twas already declared in line " +
      globals[S1->name].line + " with " +
std::to_string(globals[S1->name].arity) + " argument" +
      (globals[S1->name].arity != 1 ? "s" : "") + "!\n";
      // дано m аргумент(ов), но процедура 'f' была уже
объявлена в строке 1 с n параметром(ами)!
      // given 5 arguments, but the procedure 'f' was
already declared in line 1 with n agrument(s)!
      return 1;
   }

   params.clear();
   scope = 0;
   globals[S1->name].set(DEFINED);
   return 0;
}

int tSM::p77() { //    HPROC -> PCPAR )
   scope = 1;
   return 0;
}

int tSM::p78() { //    HPROC -> HPROC INTER
```

```
   return 0;}

int tSM::p79() { //    PCPAR -> ( define ( $id
   S1->name = S4->name;
   S1->count = 0;
   return 0;
}

int tSM::p80() { //    PCPAR -> PCPAR $id
   if (params.count(S2->name)) {
      ferror_message = "Error[80-1] in line " + S1->line + ":
in the procedure '" + S1->name +
      "', \n\t\tthe parameter named '" + S2->name + "' is
duplicated!\n";
      // в процедуре 'a' дублируется параметр с именем
'b'!
      // in the procedure 'a', the parameter named 'b' is
duplicated!
      return 1;
   }

   if (S2->name == S1->name) {
      ferror_message +=
      "Warning[80-2] in line " + S2->line + ": the procedure
'"
      + S1->name + " has the same name \n\t\t\tas its
parameter!\n";
      // процедура 'f' имеет такое же имя как параметр!
      // the procedure 'f' has the same name as its
parameter!
   }

   ++S1->count;
   params.insert(S2->name);
   return 0;
}

int tSM::p81() { //    BLOCK -> HBLOCK E )
   locals.clear();
   return 0;
}
```

```
int tSM::p82() { //   HBLOCK -> BLVAR )
   scope = 2;
   return 0;
}

int tSM::p83() { //   HBLOCK -> HBLOCK INTER
   return 0;}

int tSM::p84() { //    BLVAR -> ( let ( LOCDEF
   S1->name = S4->name;
   S1->count = 1;
   locals.insert(S1->name);
   return 0;
}
int tSM::p85() { //    BLVAR -> BLVAR LOCDEF
   if (locals.count(S2->name)) {
      ferror_message = "Error[85-1] in line " + S1->line + ": 
the local variable '" +
      S1->name + "' \n\t\tduplicates another local variable 
in the block!\n";
      // локальная переменная 'a' дублирует другую 
локальную переменную в блоке!
      // the local variable 'a' duplicates another local 
variable in the block!
      return 1;
   }
   ++S1->count;
   locals.insert(S2->name);
   return 0;;
}
int tSM::p86() { //   LOCDEF -> ( $id E )
   S1->name = S2->name;
   return 0;
}

//________________________
int tSM::p87(){return 0;} int tSM::p88(){return 0;}
int tSM::p89(){return 0;} int tSM::p90(){return 0;}
int tSM::p91(){return 0;} int tSM::p92(){return 0;}
int tSM::p93(){return 0;} int tSM::p94(){return 0;}
int tSM::p95(){return 0;} int tSM::p96(){return 0;}
int tSM::p97(){return 0;} int tSM::p98(){return 0;}
int tSM::p99(){return 0;} int tSM::p100(){return 0;}
int tSM::p101(){return 0;} int tSM::p102(){return 0;}
int tSM::p103(){return 0;} int tSM::p104(){return 0;}
int tSM::p105(){return 0;} int tSM::p106(){return 0;}
int tSM::p107(){return 0;} int tSM::p108(){return 0;}
int tSM::p109(){return 0;} int tSM::p110(){return 0;}
```