```
    1:
    2: RUNNING TEST CASE FILE "test1.txt"
    3:
    4:
    5: Token:  Keyword         Lexeme: function
    6:
    7:         <Rat18F> -> <Opt Function Definitions> $$ <Opt Declaration List> <Statemen
t List>
    8:         <Opt Function Definitions> ->  <Function Definitions> | <Empty>
    9:
   10: Token:  Identifier      Lexeme: add
   11:
   12:         <Function Definitions> -> <Function> | <Function> <Function Definitions>
   13:         <Function> ->  function <Identifier> ( <Opt Parameter List> ) <Opt Declara
tion List> <Body>
   14:         <Identifier>
   15:
   16: Token:  Separator       Lexeme: (
   17:
   18:
   19: Token:  Identifier      Lexeme: a
   20:
   21:         <Opt Parameter List> -> <Parameter List> | <Empty>
   22:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
   23:         <Parameter> -> <IDs> : <Qualifier>
   24:         <IDs> -> <Identifier> | <Identifier>, <IDs>
   25:         <Identifier>
   26:
   27: Token:  Separator       Lexeme: :
   28:
   29:
   30: Token:  Keyword         Lexeme: int
   31:
   32:         <Qualifier> ->  int | boolean | real
   33:
   34: Token:  Separator       Lexeme: ,
   35:
   36:
   37: Token:  Identifier      Lexeme: b
   38:
   39:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
   40:         <Parameter> -> <IDs> : <Qualifier>
   41:         <IDs> -> <Identifier> | <Identifier>, <IDs>
   42:         <Identifier>
   43:
   44: Token:  Separator       Lexeme: :
   45:
   46:
   47: Token:  Keyword         Lexeme: int
   48:
   49:         <Qualifier> ->  int | boolean | real
   50:
   51: Token:  Separator       Lexeme: )
   52:
   53:
   54: Token:  Separator       Lexeme: {
   55:
   56:         <Opt Declaration List> -> <Declaration List> | <Empty>
   57:         <Empty> -> Îµ
   58:         <Body> -> { <Statement List> }
   59:
   60: Token:  Keyword         Lexeme: return
   61:
   62:         <Statement List> -> <Statement> | <Statement> <Statement List>
   63:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
   64:
   65: Token:  Identifier      Lexeme: a
   66:
   67:         <Return> -> return; |  return <Expression>;
```

```
   68:         <Expression> -> <Term> <ExpressionPrime>
   69:         <Term> -> <Factor> <TermPrime>
   70:         <Factor> -> - <Primary> | <Primary>
   71:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
   72:         <Identifier>
   73:
   74: Token:  Operator        Lexeme: +
   75:
   76:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
   77:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
   78:
   79: Token:  Identifier      Lexeme: b
   80:
   81:         <Term> -> <Factor> <TermPrime>
   82:         <Factor> -> - <Primary> | <Primary>
   83:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
   84:         <Identifier>
   85:
   86: Token:  Separator       Lexeme: ;
   87:
   88:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
   89:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
   90:         <Empty> -> Îµ
   91:
   92: Token:  Separator       Lexeme: }
   93:
   94:
   95: Token:  Keyword         Lexeme: function
   96:
   97:
   98: Token:  Identifier      Lexeme: subtract
   99:
  100:         <Function Definitions> -> <Function> | <Function> <Function Definitions>
  101:         <Function> ->  function <Identifier> ( <Opt Parameter List> ) <Opt Declara
tion List> <Body>
  102:         <Identifier>
  103:
  104: Token:  Separator       Lexeme: (
  105:
  106:
  107: Token:  Identifier      Lexeme: a
  108:
  109:         <Opt Parameter List> -> <Parameter List> | <Empty>
  110:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  111:         <Parameter> -> <IDs> : <Qualifier>
  112:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  113:         <Identifier>
  114:
  115: Token:  Separator       Lexeme: :
  116:
  117:
  118: Token:  Keyword         Lexeme: int
  119:
  120:         <Qualifier> ->  int | boolean | real
  121:
  122: Token:  Separator       Lexeme: ,
  123:
  124:
  125: Token:  Identifier      Lexeme: b
  126:
  127:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  128:         <Parameter> -> <IDs> : <Qualifier>
  129:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  130:         <Identifier>
  131:
  132: Token:  Separator       Lexeme: :
```

```
  133:
  134:
  135: Token:  Keyword         Lexeme: int
  136:
  137:         <Qualifier> ->  int | boolean | real
  138:
  139: Token:  Separator       Lexeme: )
  140:
  141:
  142: Token:  Separator       Lexeme: {
  143:
  144:         <Opt Declaration List> -> <Declaration List> | <Empty>
  145:         <Empty> -> Îµ
  146:         <Body> -> { <Statement List> }
  147:
  148: Token:  Keyword         Lexeme: return
  149:
  150:         <Statement List> -> <Statement> | <Statement> <Statement List>
  151:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  152:
  153: Token:  Identifier      Lexeme: a
  154:
  155:         <Return> -> return; |  return <Expression>;
  156:         <Expression> -> <Term> <ExpressionPrime>
  157:         <Term> -> <Factor> <TermPrime>
  158:         <Factor> -> - <Primary> | <Primary>
  159:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  160:         <Identifier>
  161:
  162: Token:  Operator        Lexeme: -
  163:
  164:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  165:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  166:
  167: Token:  Identifier      Lexeme: b
  168:
  169:         <Term> -> <Factor> <TermPrime>
  170:         <Factor> -> - <Primary> | <Primary>
  171:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  172:         <Identifier>
  173:
  174: Token:  Separator       Lexeme: ;
  175:
  176:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  177:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  178:         <Empty> -> Îµ
  179:
  180: Token:  Separator       Lexeme: }
  181:
  182:
  183: Token:  Keyword         Lexeme: function
  184:
  185:
  186: Token:  Identifier      Lexeme: divide
  187:
  188:         <Function Definitions> -> <Function> | <Function> <Function Definitions>
  189:         <Function> ->  function <Identifier> ( <Opt Parameter List> ) <Opt Declara
tion List> <Body>
  190:         <Identifier>
  191:
  192: Token:  Separator       Lexeme: (
  193:
  194:
  195: Token:  Identifier      Lexeme: a
  196:
```

```
  197:         <Opt Parameter List> -> <Parameter List> | <Empty>
  198:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  199:         <Parameter> -> <IDs> : <Qualifier>
  200:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  201:         <Identifier>
  202:
  203: Token:  Separator       Lexeme: :
  204:
  205:
  206: Token:  Keyword         Lexeme: int
  207:
  208:         <Qualifier> ->  int | boolean | real
  209:
  210: Token:  Separator       Lexeme: ,
  211:
  212:
  213: Token:  Identifier      Lexeme: b
  214:
  215:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  216:         <Parameter> -> <IDs> : <Qualifier>
  217:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  218:         <Identifier>
  219:
  220: Token:  Separator       Lexeme: :
  221:
  222:
  223: Token:  Keyword         Lexeme: int
  224:
  225:         <Qualifier> ->  int | boolean | real
  226:
  227: Token:  Separator       Lexeme: )
  228:
  229:
  230: Token:  Separator       Lexeme: {
  231:
  232:         <Opt Declaration List> -> <Declaration List> | <Empty>
  233:         <Empty> -> Îµ
  234:         <Body> -> { <Statement List> }
  235:
  236: Token:  Keyword         Lexeme: return
  237:
  238:         <Statement List> -> <Statement> | <Statement> <Statement List>
  239:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  240:
  241: Token:  Identifier      Lexeme: a
  242:
  243:         <Return> -> return; |  return <Expression>;
  244:         <Expression> -> <Term> <ExpressionPrime>
  245:         <Term> -> <Factor> <TermPrime>
  246:         <Factor> -> - <Primary> | <Primary>
  247:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  248:         <Identifier>
  249:
  250: Token:  Operator        Lexeme: /
  251:
  252:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  253:
  254: Token:  Identifier      Lexeme: b
  255:
  256:         <Factor> -> - <Primary> | <Primary>
  257:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  258:         <Identifier>
  259:
  260: Token:  Separator       Lexeme: ;
  261:
  262:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  263:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
```

```
e> | <Empty>
  264:         <Empty> -> Îµ
  265:
  266: Token:  Separator       Lexeme: }
  267:
  268:
  269: Token:  Separator       Lexeme: $$
  270:
  271:
  272: Token:  Identifier      Lexeme: a
  273:
  274:         <Opt Declaration List> -> <Declaration List> | <Empty>
  275:         <Empty> -> Îµ
  276:         <Statement List> -> <Statement> | <Statement> <Statement List>
  277:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  278:         <Assign> -> <Identifier> = <Expression>;
  279:         <Identifier>
  280:
  281: Token:  Operator        Lexeme: =
  282:
  283:
  284: Token:  Integer         Lexeme: 3
  285:
  286:         <Expression> -> <Term> <ExpressionPrime>
  287:         <Term> -> <Factor> <TermPrime>
  288:         <Factor> -> - <Primary> | <Primary>
  289:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  290:         <Integer>
  291:
  292: Token:  Separator       Lexeme: ;
  293:
  294:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  295:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  296:         <Empty> -> Îµ
  297:
  298: Token:  Identifier      Lexeme: b
  299:
  300:         <Statement List> -> <Statement> | <Statement> <Statement List>
  301:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  302:         <Assign> -> <Identifier> = <Expression>;
  303:         <Identifier>
  304:
  305: Token:  Operator        Lexeme: =
  306:
  307:
  308: Token:  Integer         Lexeme: 5
  309:
  310:         <Expression> -> <Term> <ExpressionPrime>
  311:         <Term> -> <Factor> <TermPrime>
  312:         <Factor> -> - <Primary> | <Primary>
  313:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  314:         <Integer>
  315:
  316: Token:  Separator       Lexeme: ;
  317:
  318:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  319:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  320:         <Empty> -> Îµ
  321:
  322: Token:  Keyword         Lexeme: put
  323:
  324:         <Statement List> -> <Statement> | <Statement> <Statement List>
  325:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
```

```
  326:
  327: Token:  Separator       Lexeme: (
  328:
  329:         <Print> -> put ( <Expression> );
  330:
  331: Token:  Identifier      Lexeme: a
  332:
  333:         <Expression> -> <Term> <ExpressionPrime>
  334:         <Term> -> <Factor> <TermPrime>
  335:         <Factor> -> - <Primary> | <Primary>
  336:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  337:         <Identifier>
  338:
  339: Token:  Separator       Lexeme: )
  340:
  341:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  342:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  343:         <Empty> -> Îµ
  344:
  345: Token:  Separator       Lexeme: ;
  346:
  347:
  348: Token:  Identifier      Lexeme: a
  349:
  350:         <Statement List> -> <Statement> | <Statement> <Statement List>
  351:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  352:         <Assign> -> <Identifier> = <Expression>;
  353:         <Identifier>
  354:
  355: Token:  Operator        Lexeme: =
  356:
  357:
  358: Token:  Identifier      Lexeme: a
  359:
  360:         <Expression> -> <Term> <ExpressionPrime>
  361:         <Term> -> <Factor> <TermPrime>
  362:         <Factor> -> - <Primary> | <Primary>
  363:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  364:         <Identifier>
  365:
  366: Token:  Operator        Lexeme: +
  367:
  368:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  369:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  370:
  371: Token:  Identifier      Lexeme: b
  372:
  373:         <Term> -> <Factor> <TermPrime>
  374:         <Factor> -> - <Primary> | <Primary>
  375:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  376:         <Identifier>
  377:
  378: Token:  Separator       Lexeme: ;
  379:
  380:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  381:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  382:         <Empty> -> Îµ
  383:
  384: Token:  Separator       Lexeme: $$
  385:
  386: Syntax Analysis Successful.
  387:
  388: RUNNING TEST CASE FILE "test2.txt"
```

```
  389:
  390:
  391: Token:  Keyword         Lexeme: function
  392:
  393:         <Rat18F> -> <Opt Function Definitions> $$ <Opt Declaration List> <Statemen
t List>
  394:         <Opt Function Definitions> ->  <Function Definitions> | <Empty>
  395:
  396: Token:  Identifier      Lexeme: isEqual
  397:
  398:         <Function Definitions> -> <Function> | <Function> <Function Definitions>
  399:         <Function> ->  function <Identifier> ( <Opt Parameter List> ) <Opt Declara
tion List> <Body>
  400:         <Identifier>
  401:
  402: Token:  Separator       Lexeme: (
  403:
  404:
  405: Token:  Identifier      Lexeme: a
  406:
  407:         <Opt Parameter List> -> <Parameter List> | <Empty>
  408:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  409:         <Parameter> -> <IDs> : <Qualifier>
  410:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  411:         <Identifier>
  412:
  413: Token:  Separator       Lexeme: :
  414:
  415:
  416: Token:  Keyword         Lexeme: int
  417:
  418:         <Qualifier> ->  int | boolean | real
  419:
  420: Token:  Separator       Lexeme: ,
  421:
  422:
  423: Token:  Identifier      Lexeme: b
  424:
  425:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  426:         <Parameter> -> <IDs> : <Qualifier>
  427:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  428:         <Identifier>
  429:
  430: Token:  Separator       Lexeme: :
  431:
  432:
  433: Token:  Keyword         Lexeme: int
  434:
  435:         <Qualifier> ->  int | boolean | real
  436:
  437: Token:  Separator       Lexeme: )
  438:
  439:
  440: Token:  Separator       Lexeme: {
  441:
  442:         <Opt Declaration List> -> <Declaration List> | <Empty>
  443:         <Empty> -> Îµ
  444:         <Body> -> { <Statement List> }
  445:
  446: Token:  Keyword         Lexeme: return
  447:
  448:         <Statement List> -> <Statement> | <Statement> <Statement List>
  449:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  450:
  451: Token:  Integer         Lexeme: 10
  452:
  453:         <Return> -> return; |  return <Expression>;
  454:         <Expression> -> <Term> <ExpressionPrime>
  455:         <Term> -> <Factor> <TermPrime>
```

```
  456:         <Factor> -> - <Primary> | <Primary>
  457:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  458:         <Integer>
  459:
  460: Token:  Operator        Lexeme: -
  461:
  462:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  463:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  464:
  465: Token:  Identifier      Lexeme: a
  466:
  467:         <Term> -> <Factor> <TermPrime>
  468:         <Factor> -> - <Primary> | <Primary>
  469:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  470:         <Identifier>
  471:
  472: Token:  Operator        Lexeme: +
  473:
  474:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  475:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  476:
  477: Token:  Identifier      Lexeme: b
  478:
  479:         <Term> -> <Factor> <TermPrime>
  480:         <Factor> -> - <Primary> | <Primary>
  481:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  482:         <Identifier>
  483:
  484: Token:  Separator       Lexeme: ;
  485:
  486:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  487:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  488:         <Empty> -> Îµ
  489:
  490: Token:  Separator       Lexeme: }
  491:
  492:
  493: Token:  Keyword         Lexeme: function
  494:
  495:
  496: Token:  Identifier      Lexeme: convert1x
  497:
  498:         <Function Definitions> -> <Function> | <Function> <Function Definitions>
  499:         <Function> ->  function <Identifier> ( <Opt Parameter List> ) <Opt Declara
tion List> <Body>
  500:         <Identifier>
  501:
  502: Token:  Separator       Lexeme: (
  503:
  504:
  505: Token:  Identifier      Lexeme: fahr
  506:
  507:         <Opt Parameter List> -> <Parameter List> | <Empty>
  508:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  509:         <Parameter> -> <IDs> : <Qualifier>
  510:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  511:         <Identifier>
  512:
  513: Token:  Separator       Lexeme: :
  514:
  515:
  516: Token:  Keyword         Lexeme: int
  517:
  518:         <Qualifier> ->  int | boolean | real
```

```
  519:
  520: Token:  Separator       Lexeme: )
  521:
  522:
  523: Token:  Separator       Lexeme: {
  524:
  525:         <Opt Declaration List> -> <Declaration List> | <Empty>
  526:         <Empty> -> Îµ
  527:         <Body> -> { <Statement List> }
  528:
  529: Token:  Keyword         Lexeme: return
  530:
  531:         <Statement List> -> <Statement> | <Statement> <Statement List>
  532:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  533:
  534: Token:  Integer         Lexeme: 5
  535:
  536:         <Return> -> return; |  return <Expression>;
  537:         <Expression> -> <Term> <ExpressionPrime>
  538:         <Term> -> <Factor> <TermPrime>
  539:         <Factor> -> - <Primary> | <Primary>
  540:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  541:         <Integer>
  542:
  543: Token:  Operator        Lexeme: *
  544:
  545:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  546:
  547: Token:  Separator       Lexeme: (
  548:
  549:         <Factor> -> - <Primary> | <Primary>
  550:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  551:
  552: Token:  Identifier      Lexeme: fahr
  553:
  554:         <Expression> -> <Term> <ExpressionPrime>
  555:         <Term> -> <Factor> <TermPrime>
  556:         <Factor> -> - <Primary> | <Primary>
  557:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  558:         <Identifier>
  559:
  560: Token:  Operator        Lexeme: -
  561:
  562:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  563:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  564:
  565: Token:  Integer         Lexeme: 32
  566:
  567:         <Term> -> <Factor> <TermPrime>
  568:         <Factor> -> - <Primary> | <Primary>
  569:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  570:         <Integer>
  571:
  572: Token:  Separator       Lexeme: )
  573:
  574:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  575:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  576:         <Empty> -> Îµ
  577:
  578: Token:  Operator        Lexeme: /
  579:
  580:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  581:
```

```
  582: Token:  Integer         Lexeme: 9
  583:
  584:         <Factor> -> - <Primary> | <Primary>
  585:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  586:         <Integer>
  587:
  588: Token:  Separator       Lexeme: ;
  589:
  590:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  591:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  592:         <Empty> -> Îµ
  593:
  594: Token:  Separator       Lexeme: }
  595:
  596:
  597: Token:  Separator       Lexeme: $$
  598:
  599:
  600: Token:  Keyword         Lexeme: int
  601:
  602:         <Opt Declaration List> -> <Declaration List> | <Empty>
  603:         <Declaration List> -> <Declaration>; | <Declaration>; <Declaration List>
  604:         <Declaration> -> <Qualifier> <IDs>
  605:         <Qualifier> ->  int | boolean | real
  606:
  607: Token:  Identifier      Lexeme: low
  608:
  609:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  610:         <Identifier>
  611:
  612: Token:  Separator       Lexeme: ,
  613:
  614:
  615: Token:  Identifier      Lexeme: high
  616:
  617:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  618:         <Identifier>
  619:
  620: Token:  Separator       Lexeme: ,
  621:
  622:
  623: Token:  Identifier      Lexeme: step
  624:
  625:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  626:         <Identifier>
  627:
  628: Token:  Separator       Lexeme: ;
  629:
  630:
  631: Token:  Keyword         Lexeme: get
  632:
  633:         <Statement List> -> <Statement> | <Statement> <Statement List>
  634:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  635:
  636: Token:  Separator       Lexeme: (
  637:
  638:         <Scan> -> get ( <IDs> );
  639:
  640: Token:  Identifier      Lexeme: low
  641:
  642:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  643:         <Identifier>
  644:
  645: Token:  Separator       Lexeme: ,
  646:
  647:
  648: Token:  Identifier      Lexeme: high
```

```
  649:
  650:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  651:         <Identifier>
  652:
  653: Token:  Separator       Lexeme: ,
  654:
  655:
  656: Token:  Identifier      Lexeme: step
  657:
  658:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  659:         <Identifier>
  660:
  661: Token:  Separator       Lexeme: )
  662:
  663:
  664: Token:  Separator       Lexeme: ;
  665:
  666:
  667: Token:  Keyword         Lexeme: while
  668:
  669:         <Statement List> -> <Statement> | <Statement> <Statement List>
  670:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  671:
  672: Token:  Separator       Lexeme: (
  673:
  674:         <While> -> while ( <Condition> )  <Statement>
  675:
  676: Token:  Identifier      Lexeme: low
  677:
  678:         <Condition> -> <Expression>  <Relop>  <Expression>
  679:         <Expression> -> <Term> <ExpressionPrime>
  680:         <Term> -> <Factor> <TermPrime>
  681:         <Factor> -> - <Primary> | <Primary>
  682:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  683:         <Identifier>
  684:
  685: Token:  Operator        Lexeme: <
  686:
  687:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  688:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  689:         <Empty> -> Îµ
  690:         <Relop> -> <
  691:
  692: Token:  Identifier      Lexeme: high
  693:
  694:         <Expression> -> <Term> <ExpressionPrime>
  695:         <Term> -> <Factor> <TermPrime>
  696:         <Factor> -> - <Primary> | <Primary>
  697:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  698:         <Identifier>
  699:
  700: Token:  Separator       Lexeme: )
  701:
  702:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  703:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  704:         <Empty> -> Îµ
  705:
  706: Token:  Separator       Lexeme: {
  707:
  708:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  709:
  710: Token:  Keyword         Lexeme: put
  711:
  712:         <Compound> -> { <Statement List> }
```

```
  713:         <Statement List> -> <Statement> | <Statement> <Statement List>
  714:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  715:
  716: Token:  Separator       Lexeme: (
  717:
  718:         <Print> -> put ( <Expression> );
  719:
  720: Token:  Identifier      Lexeme: low
  721:
  722:         <Expression> -> <Term> <ExpressionPrime>
  723:         <Term> -> <Factor> <TermPrime>
  724:         <Factor> -> - <Primary> | <Primary>
  725:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  726:         <Identifier>
  727:
  728: Token:  Separator       Lexeme: )
  729:
  730:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  731:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  732:         <Empty> -> Îµ
  733:
  734: Token:  Separator       Lexeme: ;
  735:
  736:
  737: Token:  Keyword         Lexeme: put
  738:
  739:         <Statement List> -> <Statement> | <Statement> <Statement List>
  740:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  741:
  742: Token:  Separator       Lexeme: (
  743:
  744:         <Print> -> put ( <Expression> );
  745:
  746: Token:  Identifier      Lexeme: convert1x
  747:
  748:         <Expression> -> <Term> <ExpressionPrime>
  749:         <Term> -> <Factor> <TermPrime>
  750:         <Factor> -> - <Primary> | <Primary>
  751:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  752:         <Identifier>
  753:
  754: Token:  Separator       Lexeme: (
  755:
  756:
  757: Token:  Identifier      Lexeme: low
  758:
  759:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  760:         <Identifier>
  761:
  762: Token:  Separator       Lexeme: )
  763:
  764:
  765: Token:  Separator       Lexeme: )
  766:
  767:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  768:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  769:         <Empty> -> Îµ
  770:
  771: Token:  Separator       Lexeme: ;
  772:
  773:
  774: Token:  Identifier      Lexeme: low
  775:
  776:         <Statement List> -> <Statement> | <Statement> <Statement List>
```

```
  777:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  778:         <Assign> -> <Identifier> = <Expression>;
  779:         <Identifier>
  780:
  781: Token:  Operator        Lexeme: =
  782:
  783:
  784: Token:  Identifier      Lexeme: low
  785:
  786:         <Expression> -> <Term> <ExpressionPrime>
  787:         <Term> -> <Factor> <TermPrime>
  788:         <Factor> -> - <Primary> | <Primary>
  789:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  790:         <Identifier>
  791:
  792: Token:  Operator        Lexeme: +
  793:
  794:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  795:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  796:
  797: Token:  Identifier      Lexeme: step
  798:
  799:         <Term> -> <Factor> <TermPrime>
  800:         <Factor> -> - <Primary> | <Primary>
  801:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  802:         <Identifier>
  803:
  804: Token:  Separator       Lexeme: ;
  805:
  806:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  807:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  808:         <Empty> -> Îµ
  809:
  810: Token:  Separator       Lexeme: }
  811:
  812:
  813: Token:  Keyword         Lexeme: whileend
  814:
  815:
  816: Token:  Separator       Lexeme: $$
  817:
  818: Syntax Analysis Successful.
  819:
  820: RUNNING TEST CASE FILE "test3.txt"
  821:
  822:
  823: Token:  Keyword         Lexeme: function
  824:
  825:         <Rat18F> -> <Opt Function Definitions> $$ <Opt Declaration List> <Statemen
t List>
  826:         <Opt Function Definitions> ->  <Function Definitions> | <Empty>
  827:
  828: Token:  Identifier      Lexeme: myInvalidFunction
  829:
  830:         <Function Definitions> -> <Function> | <Function> <Function Definitions>
  831:         <Function> ->  function <Identifier> ( <Opt Parameter List> ) <Opt Declara
tion List> <Body>
  832:         <Identifier>
  833:
  834: Token:  Separator       Lexeme: (
  835:
  836:
  837: Token:  Identifier      Lexeme: var1x
  838:
  839:         <Opt Parameter List> -> <Parameter List> | <Empty>
```

```
  840:         <Parameter List> -> <Parameter> | <Parameter> , <Parameter List>
  841:         <Parameter> -> <IDs> : <Qualifier>
  842:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  843:         <Identifier>
  844:
  845: Token:  Separator       Lexeme: :
  846:
  847:
  848: Token:  Keyword         Lexeme: int
  849:
  850:         <Qualifier> ->  int | boolean | real
  851:
  852: Token:  Separator       Lexeme: )
  853:
  854:
  855: Token:  Separator       Lexeme: {
  856:
  857:         <Opt Declaration List> -> <Declaration List> | <Empty>
  858:         <Empty> -> Îµ
  859:         <Body> -> { <Statement List> }
  860:
  861: Token:  Keyword         Lexeme: return
  862:
  863:         <Statement List> -> <Statement> | <Statement> <Statement List>
  864:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  865:
  866: Token:  Separator       Lexeme: (
  867:
  868:         <Return> -> return; |  return <Expression>;
  869:         <Expression> -> <Term> <ExpressionPrime>
  870:         <Term> -> <Factor> <TermPrime>
  871:         <Factor> -> - <Primary> | <Primary>
  872:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  873:
  874: Token:  Identifier      Lexeme: var1x
  875:
  876:         <Expression> -> <Term> <ExpressionPrime>
  877:         <Term> -> <Factor> <TermPrime>
  878:         <Factor> -> - <Primary> | <Primary>
  879:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  880:         <Identifier>
  881:
  882: Token:  Operator        Lexeme: *
  883:
  884:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  885:
  886: Token:  Identifier      Lexeme: var2x
  887:
  888:         <Factor> -> - <Primary> | <Primary>
  889:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  890:         <Identifier>
  891:
  892: Token:  Separator       Lexeme: )
  893:
  894:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  895:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  896:         <Empty> -> Îµ
  897:
  898: Token:  Operator        Lexeme: +
  899:
  900:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  901:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  902:
  903: Token:  Integer         Lexeme: 50
```

```
  904:
  905:         <Term> -> <Factor> <TermPrime>
  906:         <Factor> -> - <Primary> | <Primary>
  907:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  908:         <Integer>
  909:
  910: Token:  Separator       Lexeme: ;
  911:
  912:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  913:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  914:         <Empty> -> Îµ
  915:
  916: Token:  Separator       Lexeme: }
  917:
  918:
  919: Token:  Separator       Lexeme: $$
  920:
  921:
  922: Token:  Keyword         Lexeme: int
  923:
  924:         <Opt Declaration List> -> <Declaration List> | <Empty>
  925:         <Declaration List> -> <Declaration>; | <Declaration>; <Declaration List>
  926:         <Declaration> -> <Qualifier> <IDs>
  927:         <Qualifier> ->  int | boolean | real
  928:
  929: Token:  Identifier      Lexeme: a
  930:
  931:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  932:         <Identifier>
  933:
  934: Token:  Separator       Lexeme: ,
  935:
  936:
  937: Token:  Identifier      Lexeme: b
  938:
  939:         <IDs> -> <Identifier> | <Identifier>, <IDs>
  940:         <Identifier>
  941:
  942: Token:  Separator       Lexeme: ;
  943:
  944:
  945: Token:  Identifier      Lexeme: a
  946:
  947:         <Statement List> -> <Statement> | <Statement> <Statement List>
  948:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  949:         <Assign> -> <Identifier> = <Expression>;
  950:         <Identifier>
  951:
  952: Token:  Operator        Lexeme: =
  953:
  954:
  955: Token:  Integer         Lexeme: 0
  956:
  957:         <Expression> -> <Term> <ExpressionPrime>
  958:         <Term> -> <Factor> <TermPrime>
  959:         <Factor> -> - <Primary> | <Primary>
  960:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  961:         <Integer>
  962:
  963: Token:  Separator       Lexeme: ;
  964:
  965:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  966:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  967:         <Empty> -> Îµ
  968:
```

```
  969: Token:  Identifier      Lexeme: b
  970:
  971:         <Statement List> -> <Statement> | <Statement> <Statement List>
  972:         <Statement> -> <Compound> | <Assign> | <If> | <Return> | <Print> | <Scan>
| <While>
  973:         <Assign> -> <Identifier> = <Expression>;
  974:         <Identifier>
  975:
  976: Token:  Operator        Lexeme: =
  977:
  978:
  979: Token:  Integer         Lexeme: 1
  980:
  981:         <Expression> -> <Term> <ExpressionPrime>
  982:         <Term> -> <Factor> <TermPrime>
  983:         <Factor> -> - <Primary> | <Primary>
  984:         <Primary> -> <Identifier> | <Integer> | <Identifier> ( <IDs> ) | ( <Expres
sion> ) | <Real> | true | false
  985:         <Integer>
  986:
  987: Token:  Separator       Lexeme: ;
  988:
  989:         <TermPrime> -> * <Factor> <TermPrime> | / <Factor> <TermPrime> | <Empty>
  990:         <ExpressionPrime> -> + <Term> <ExpressionPrime> | - <Term> <ExpressionPrim
e> | <Empty>
  991:         <Empty> -> Îµ
  992:
  993: Token:  Illegal         Lexeme: invalidVar1
  994:
  995:
  996: ERROR: Illegal symbol ’invalidVar1’ Line: 15
```