# Computer algebra independent integration tests

Nasser M. Abbasi

December 16, 2018 Compiled on December 16, 2018 at 10:04am

## Contents



# 1 Introduction

This report gives the result of running the computer algebra independent integration problems.

The listing of the problems used by this report are

1. MathematicaSyntaxTestSuite.zip
2. MapleSyntaxTestSuite.zip

The above zip files are maintained by and were downloaded from Albert Rich Rubi web site.

The current number of problems in this test suite is [28425].

## 1.1 Listing of CAS systems tested

The following systems were tested at this time.

1. Mathematica 11.3 (64 bit).
2. Rubi 4.15.2 in Mathematica 11.3.
3. Rubi in Sympy (Version 1.3) under Python 3.7.0 using Anaconda distribution.

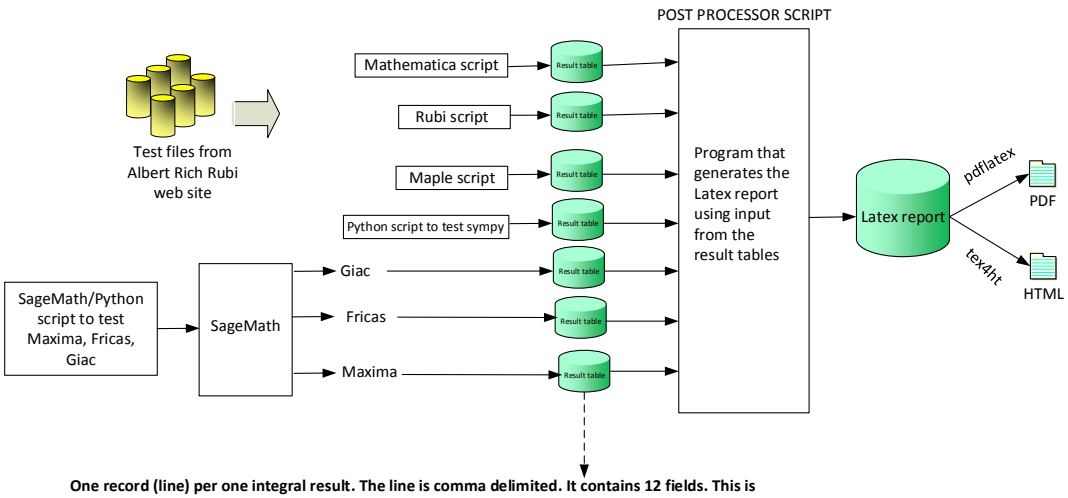
4. Maple 2018.1 (64 bit).
5. Maxima 5.41 Using Lisp ECL 16.1.2.
6. Fricas 1.3.4.
7. Sympy 1.3 under Python 3.7.0 using Anaconda distribution.
8. Giac/Xcas 1.4.9.

Maxima, Fricas and Giac/Xcas were called from inside SageMath version 8.3. This was done using SageMath integrate command by changing the name of the algorithm to use the different CAS systems.

Sympy was called directly using Python. Rubi in Sympy was also called directly using sympy 1.3 in python.

## 1.2 Design of the test system

The following diagram gives a high level view of the current test build system.




**One record (line) per one integral result. The line is comma delimited. It contains 12 fields. This is description of each record (line)**

1. integer, the problem number.
2. integer. 0 or 1 for failed or passed. (this is not the grade field)
3. integer. Leaf size of result.
4. integer. Leaf size of the optimal antiderivative.
5. number. CPU time used to solve this integral. 0 if failed.
6. string. The integral in Latex format
7. string. The input used in CAS own syntx.
8. string. The result (antiderivative) produced by CAS in Latex format
9. string. The optimal antiderivative in Latex format.
10. integer. 0 or 1. Indicates if problem has known antiderivative or not
11. String. The result (antiderivative) in CAS own syntax.
12. String. The grade of the antiderivative. Can be “A”, “B”, “C”, or “F”

**High level overview of the CAS independent integration test build system**

Nasser M. Abbasi
June 22, 2018

## 1.3 Timing

The command `AboluteTiming[]` was used in Mathematica to obtain the elapsed time for each integrate call. In Maple, the command `Usage` was used as in the following example

```
cpu_time := Usage(assign ('result_of _int',int(expr,x)),output='realtime'
```

For all other CAS systems, the elapsed time to complete each integral was found by taking the difference between the time after the call has completed from the time before the call was made. This was done using Python's `time.time()` call.

All elapsed times shown are in seconds. A time limit of 3 minutes was used for each integral. If the integrate command did not complete within this time limit, the integral was aborted and considered to have failed and assigned an F grade. The time used by failed integrals due to time out is not counted in the final statistics.

## 1.4 Verification

A verification phase was applied on the result of integration for Rubi and Mathematica. Future version of this report will implement verification for the other CAS systems. For the integrals whose result was not run through a verification phase, it is assumed that the antiderivative produced was correct.

Verification phase has 3 minutes time out. An integral whose result was not verified could still be correct. Further investigation is needed on those integrals which failed verifications. Such integrals are marked in the summary table below and also in each integral separate section so they are easy to identify and locate.

## 1.5 Important notes about some of the results

Important note about Maxima results Since these integrals are run in a batch mode, using an automated script, and by using sagemath (SageMath uses Maxima), then any integral where Maxima needs an interactive response from the user to answer a question during evaluation of the integral in order to complete the integration, will fail and is counted as failed.

The exception raised is `ValueError`. Therefore Maxima result below is lower than what could result if Maxima was run directly and each question Maxima asks was answered correctly.

The percentage of such failures were not counted for each test file, but for an example, for the Timofeev test file, there were about 30 such integrals out of total 705, or about 4 percent. This pecrentage can be higher or lower depending on the specific input test file.

Such integrals can be indentified by looking at the output of the integration in each section for Maxima. If the output was an exception `ValueError` then this is most likely due to this reason.

Maxima integrate was run using SageMath with the following settings set by default

```
'besselexpand : true'
'display2d : false'
'domain : complex'
'keepfloat : true'
'load(to_poly_solve)'
'load(simplify_sum)'
'load(abs_integrate)' 'load(diag)'
```

SageMath loading of Maxima `abs_integrate` was found to cause some problem. So the following code was added to disable this effect.

```
from sage.interfaces.maxima_lib import maxima_lib
maxima_lib.set('extra_definite_integration_methods', '[]')
maxima_lib.set('extra_integration_methods', '[]')
```

See `https://ask.sagemath.org/question/43088/integrate-results-that-are-different-from-using-maxima/` for reference.

Important note about FriCAS and Giac/XCAS results There are Few integrals which failed due to SageMath not able to translate the result back to SageMath syntax and not because these CAS system were not able to do the integrations.

These will fail With error `Exception raised: NotImplementedError`

The number of such cases seems to be very small. About 1 or 2 percent of all integrals.

Hopefully the next version of SageMath will have complete translation of FriCAS and XCAS syntax and I will re-run all the tests again when this happens.

Important note about finding leaf size of antiderivative For Mathematica, Rubi and Maple, the buildin system function `LeafSize` is used to find the leaf size of each antiderivative.

The other CAS systems (SageMath and Sympy) do not have special buildin function for this purpose at this time. Therefore the leaf size is determined as follows.

For Fricas, Giac and Maxima (all called via sagemath) the following code is used

```
#see https://stackoverflow.com/questions/25202346/how-to-obtain-leaf-count-expressi

def tree(expr):
    if expr.operator() is None:
       return expr
    else:
       return [expr.operator()]+map(tree, expr.operands())

try:
    # 1.35 is a fudge factor since this estimate of leaf count is bit lower than
    #what it should be compared to Mathematica's
    leafCount = round(1.35*len(flatten(tree(anti))))
except Exception as ee:
    leafCount =1
```

For Sympy, called directly from Python, the following code is used

```
try:
  # 1.7 is a fudge factor since it is low side from actual leaf count
  leafCount = round(1.7*count_ops(anti))

  except Exception as ee:
         leafCount =1
```

When these cas systems implement a buildin function to find the leaf size of expressions, it will be used instead, and these tests run again.

## 1.6 Grading of results

The table below summarizes the grading of each CAS system.

Important note: A number of problems in this test suite have no antiderivative in closed form. This means the antiderivative of these integrals can not be expressed in terms of elementary, special functions or `Hypergeometric2F1` functions. `RootSum` and `RootOf` are not allowed.

If a CAS returns the above integral unevaluated within the time limit, then the result is counted as passed and assigned an A grade.

However, if CAS times out, then it is assigned an F grade even if the integral is not integrable, as this implies CAS could not determine that the integral is not integrable in the time limit.

If a CAS returns an antiderivative to such an integral, it is assigned an A grade automatically and this special result is listed in the introduction section of each individual test report to make it easy to identify as this can be important result to investigate.

The results given in in the table below reflects the above.

| System | solved | Failed |
|---|---|---|
| Rubi | % 99.87 ( 28387 ) | % 0.13 ( 38 ) |
| Rubi in Sympy | % 81.49 ( 23163 ) | % 18.51 ( 5262 ) |
| Mathematica | % 98.42 ( 27975 ) | % 1.58 ( 450 ) |
| Maple | % 86.91 ( 24705 ) | % 13.09 ( 3720 ) |
| Maxima | % 45.79 ( 13016 ) | % 54.21 ( 15409 ) |
| Fricas | % 75.79 ( 21544 ) | % 24.21 ( 6881 ) |
| Sympy | % 49.64 ( 14111 ) | % 50.36 ( 14314 ) |
| Giac | % 64.62 ( 18368 ) | % 35.38 ( 10057 ) |

Table 1: Time and leaf size performance for each CAS

The table below gives additional break down of the grading of quality of the antiderivatives generated by each CAS. The grading is given using the letters A,B,C and F with A being the best quality. The

grading is accomplished by comparing the antiderivative generated with the optimal antiderivatives included in the test suite. The following table describes the meaning of these grades.

| grade | description |
|---|---|
| A | Integral was solved and antiderivative is optimal in quality and leaf size. |
| B | Integral was solved and antiderivative is optimal in quality but leaf size is larger than twice the optimal antiderivatives leaf size. |
| C | Integral was solved and antiderivative is non-optimal in quality. This can be due to one or more of the following reasons<br>1. antiderivative contains a hypergeometric function and the optimal antiderivative does not.<br>2. antiderivative contains a special function and the optimal antiderivative does not.<br>3. antiderivative contains the imaginary unit and the optimal antiderivative does not. |
| F | Integral was not solved. Either the integral was returned unevaluated within the time limit, or it timed out, or CAS hanged or crashed or an exception was raised. |

Table 2: Description of grading applied to integration result

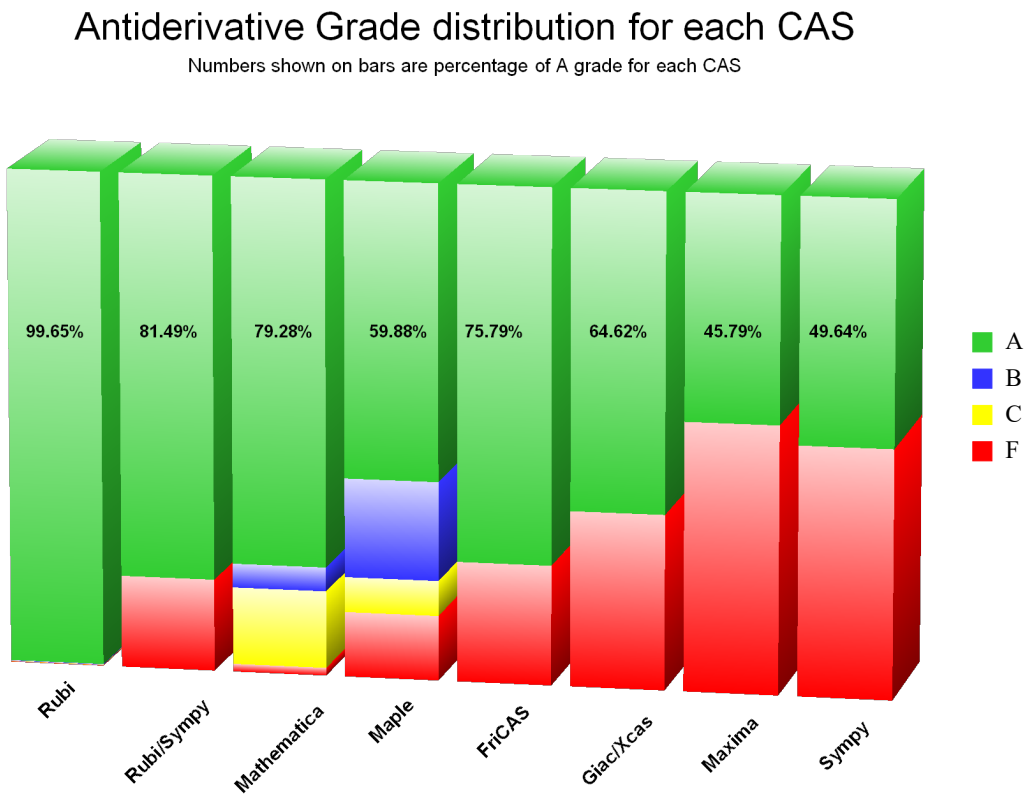
Currently grading is implemented only for Mathematica, Rubi and Maple. For other CAS systems (Maxima, Fricas, Sympy, Giac), the grading function is not yet implemented. For these systems, a grade of A is assigned if the integrate command completes successfully and a grade of F otherwise.

Based on the above, the following table summarizes the grading for this test suite.

| System | % A grade | % B grade | % C grade | % F grade |
|---|---|---|---|---|
| Rubi | 99.65 | 0.14 | 0.08 | 0.13 |
| Rubi in Sympy | 81.49 | 0. | 0. | 18.51 |
| Mathematica | 79.28 | 4.9 | 15.8 | 1.58 |
| Maple | 59.88 | 19.95 | 7.09 | 13.09 |
| Maxima | 45.79 | 0. | 0. | 54.21 |
| Fricas | 75.79 | 0. | 0. | 24.21 |
| Sympy | 49.64 | 0. | 0. | 50.36 |
| Giac | 64.62 | 0. | 0. | 35.38 |

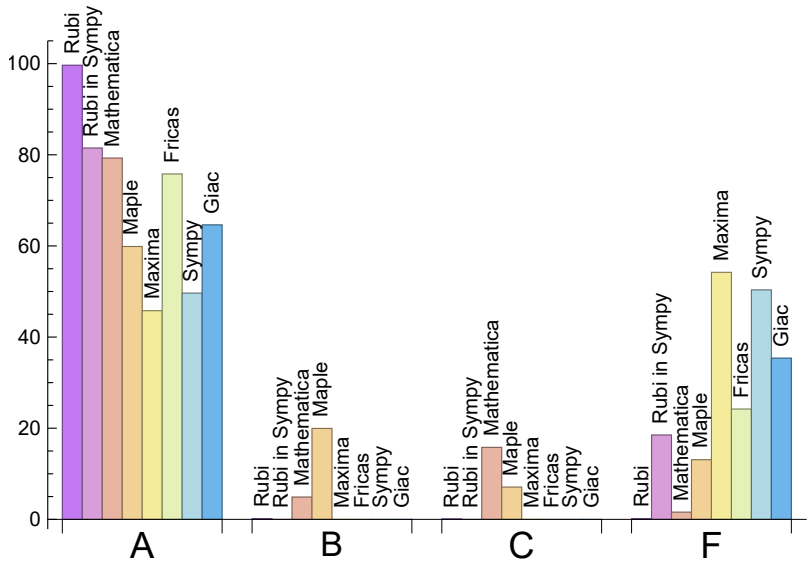
Table 3: Antiderivative Grade distribution for each CAS

The following is a Bar chart illustration of the data in the above table.




The Mathematica code used to generate the above figure is contained in this small notebook `make_3d_barchart.nb`

The figure below compares the CAS systems for each grade level.




## 1.7 Time and leaf size Performance

The table below summarizes the performance of each CAS system in terms of time used and leaf size of results.

| System | Mean time (sec) | Mean size | Normalized mean | Median size | Normalized median |
|---|---|---|---|---|---|
| Rubi | 0.44 | 147.72 | 1. | 97. | 1. |
| Rubi in Sympy | 31.17 | 116.78 | 0.9 | 82. | 0.9 |
| Mathematica | 0.56 | 229.36 | 1.3 | 85. | 0.94 |
| Maple | 0.03 | 958.29 | 4.7 | 94. | 1.05 |
| Maxima | 1.19 | 147.14 | 1.83 | 72. | 1.23 |
| Fricas | 0.89 | 278.54 | 1.86 | 58. | 1.21 |
| Sympy | 12.07 | 244.15 | 2.65 | 70. | 1.07 |
| Giac | 0.46 | 157.6 | 1.66 | 89. | 1.29 |

## 1.8 Maximum leaf size ratio for each CAS against the optimal result

The following table gives the largest ratio found in each test file, between each CAS antiderivative and the optimal antiderivative.

For each test input file, the problem with the largest ratio $\frac{\text{CAS leaf size}}{\text{Optimal leaf size}}$ is recorded with the corresponding problem number.

In each column in the table below, the first number is the maximum leaf size ratio, and the number that follows inside the parentheses is the problem number in that specific file where this maximum ratio was found. This ratio is determined only when CAS solved the the problem and also when an optimal antiderivative is known.

If it happens that a CAS was not able to solve all the integrals in the input test file, or if it was not possible to obtain leaf size for the CAS result for all the problems in the file, then a zero is used for the ratio and -1 is used for the problem number.

This makes it easy to locate the problem. In the future, a direct link will be added as well.

Table 4: Maximum leaf size ratio for each CAS against the optimal result

| file # | Rubi | Rubi in Sympy | Mathematica | Maple | Maxima | FriCAS | Sympy | Giac |
|---|---|---|---|---|---|---|---|---|
| 1 | 1. (1) | 2.2 (88) | 3.9 (50) | 16.6 (114) | 5.1 (169) | 7.2 (47) | 11.6 (144) | 15.5 (55) |
| 2 | 7.3 (21) | 2.6 (4) | 7.7 (14) | 3.6 (17) | 2.6 (4) | 13.6 (24) | 2.8 (1) | 6.2 (2) |
| 3 | 1. (1) | 1. (2) | 15.7 (6) | 16.5 (6) | 14.9 (7) | 2.4 (5) | 1.9 (5) | 2.6 (5) |
| 4 | 6.8 (5) | 2.1 (44) | 149.4 (44) | 40.8 (46) | 5.1 (40) | 11.6 (8) | 4.4 (40) | 7.2 (1) |
| 5 | 1. (38) | 5.7 (247) | 54.7 (278) | 12737.8 (278) | 11. (280) | 10.4 (280) | 62. (12) | 22.9 (161) |
| 6 | 1. (1) | 1. (1) | 1. (2) | 2.2 (4) | 2.6 (1) | 1.9 (7) | 0.8 (6) | 3.2 (5) |
| 7 | 2.2 (3) | 1. (7) | 5.6 (7) | 1.8 (3) | 3.8 (3) | 3.4 (2) | 1.9 (7) | 2.6 (3) |
| 8 | 1.6 (50) | 2.2 (33) | 5.3 (31) | 7.9 (70) | 9. (11) | 7.7 (42) | 26.4 (71) | 7. (70) |
| 9 | 1. (1) | 5.3 (301) | 7.2 (80) | 4.3 (341) | 16.3 (328) | 7.2 (142) | 8. (75) | 48.1 (368) |
| 10 | 3.2 (335) | 5.7 (573) | 242.6 (327) | 3343.5 (327) | 125.1 (408) | 106.2 (593) | 76.9 (217) | 20. (494) |
| 11 | 579. (80) | 16.1 (83) | 100. (80) | 317. (80) | 3.7 (2) | 95. (80) | 41.3 (17) | 3.7 (2) |
| 12 | 1.8 (6) | 1.3 (6) | 2.3 (4) | 1.2 (8) | 2. (2) | 2.4 (2) | 3.4 (3) | 2.1 (2) |
| 13 | 7.1 (369) | 5.6 (369) | 23.8 (1323) | 30.9 (1323) | 44.4 (1323) | 44.4 (1323) | 166. (484) | 29.8 (1488) |
| 14 | 2. (858) | 4.4 (2832) | 198.6 (3104) | 22.6 (1084) | 30. (1699) | 29.4 (1084) | 84.5 (366) | 13.7 (141) |
| 15 | 3.3 (97) | 6.6 (97) | 56. (147) | 28.5 (100) | 2.7 (154) | 9.2 (131) | 49.2 (119) | 2.6 (27) |
| 16 | 2.6 (69) | 1.7 (69) | 28.1 (136) | 68. (142) | 2.7 (69) | 11.5 (25) | 58.9 (27) | 8.7 (28) |
| 17 | 8.2 (664) | 6.5 (664) | 8.7 (663) | 7.9 (196) | 13.5 (196) | 13.5 (196) | 55.3 (528) | 14.8 (416) |
| 18 | 1.6 (251) | 3. (310) | 11.2 (346) | 147.4 (69) | 5.7 (73) | 11.9 (221) | 18.8 (32) | 6.4 (73) |
| 19 | 1. (656) | 2.8 (1050) | 3.3 (972) | 46.7 (754) | 4.2 (313) | 12. (491) | 28.8 (325) | 11.6 (553) |
| 20 | 1.3 (64) | 3.9 (106) | 3.4 (88) | 15.2 (57) | 1.5 (18) | 0. (-1) | 3. (21) | 3.2 (63) |
| 21 | 1. (1) | 1.1 (16) | 2.8 (47) | 10.4 (15) | 0. (-1) | 9.5 (15) | 43. (16) | 8.4 (18) |
| 22 | 1.2 (173) | 1.5 (163) | 1.9 (45) | 2. (162) | 3.4 (163) | 4.6 (43) | 18.3 (93) | 5.3 (157) |
| 23 | 8.4 (2679) | 6.7 (2679) | 13.4 (2906) | 141.8 (2906) | 17.8 (2278) | 15.2 (2584) | 84.8 (1709) | 17. (2807) |
| 24 | 1.4 (243) | 1.5 (273) | 42.1 (30) | 17.9 (166) | 2.6 (253) | 13.8 (106) | 28.1 (139) | 8.1 (185) |
| 25 | 1.4 (629) | 1.5 (864) | 12.3 (882) | 77.4 (546) | 39.3 (894) | 25.4 (317) | 48.4 (810) | 14.9 (882) |
| 26 | 1.2 (46) | 1.6 (26) | 6. (14) | 51.1 (15) | 0. (-1) | 38.3 (15) | 0. (-1) | 7.8 (4) |
| 27 | 1.2 (316) | 1.8 (103) | 3.8 (45) | 10.4 (43) | 4.3 (161) | 31.8 (156) | 14.9 (565) | 12.6 (579) |
| 28 | 1.3 (278) | 9.2 (331) | 10. (328) | 51.5 (297) | 11.3 (328) | 13.8 (348) | 10. (328) | 13.8 (348) |
| 29 | 1. (1) | 1.4 (40) | 4.5 (277) | 4.9 (269) | 3.1 (279) | 5.4 (269) | 21.6 (269) | 9.6 (269) |

Continued on next page

Table 4 – continued from previous page

| file # | Rubi | Rubi in Sympy | Mathematica | Maple | Maxima | FriCAS | Sympy | Giac |
|---|---|---|---|---|---|---|---|---|
| 30 | 2.8 (83) | 2.6 (124) | 4.8 (25) | 5.8 (74) | 3.5 (98) | 4.7 (103) | 16.4 (63) | 4.5 (74) |
| 31 | 2. (2408) | 4. (966) | 14.9 (2447) | 70.8 (2340) | 19.2 (1486) | 48.8 (2282) | 61.9 (1721) | 49.7 (971) |
| 32 | 1.3 (1471) | 1.5 (1471) | 16.3 (1088) | 91. (1626) | 29. (2015) | 61.5 (1452) | 48.8 (1820) | 17.7 (2163) |
| 33 | 2.1 (833) | 2.1 (833) | 46.1 (890) | 116.1 (801) | 8.1 (579) | 45.9 (616) | 71.5 (920) | 9. (357) |
| 34 | 1. (1) | 1.7 (95) | 11.1 (14) | 424.6 (78) | 3.7 (95) | 40.7 (112) | 1.2 (19) | 5.4 (121) |
| 35 | 1. (129) | 1.5 (24) | 62.2 (37) | 14197.2 (12) | 8.9 (27) | 40.7 (117) | 6.9 (13) | 7.8 (37) |
| 36 | 1.8 (76) | 2.4 (58) | 18.1 (265) | 421. (278) | 120.1 (278) | 11. (367) | 114.1 (278) | 17.2 (41) |
| 37 | 1.7 (636) | 1.7 (23) | 11.8 (1093) | 13.9 (882) | 7.3 (515) | 33.2 (1077) | 28.5 (1105) | 8.5 (785) |
| 38 | 1.7 (216) | 11. (233) | 18.1 (4) | 50.7 (224) | 9. (96) | 72.9 (117) | 19.6 (222) | 10. (96) |
| 39 | 1.9 (319) | 1.7 (319) | 15.1 (208) | 25.9 (136) | 7.5 (70) | 57.4 (388) | 47.5 (220) | 7.5 (70) |
| 40 | 1. (59) | 1.1 (95) | 3.6 (103) | 31.6 (57) | 1.8 (111) | 3.5 (46) | 43. (11) | 5. (108) |
| 41 | 1.6 (135) | 1.5 (119) | 1.8 (51) | 18.6 (128) | 2.1 (131) | 65. (60) | 27.3 (39) | 4.3 (131) |
| 42 | 1. (1) | 5. (7) | 3. (5) | 2.4 (6) | 0. (-1) | 5.6 (5) | 0. (-1) | 1.6 (13) |
| 43 | 2.1 (154) | 1.9 (362) | 32.7 (602) | 54.7 (609) | 8.5 (609) | 28.6 (632) | 26.3 (438) | 13.2 (598) |
| 44 | 1. (1) | 1.8 (69) | 36. (86) | 2.7 (37) | 1.8 (26) | 41.1 (41) | 23.7 (67) | 9.1 (67) |
| 45 | 1. (67) | 2.4 (124) | 25.1 (143) | 2909.3 (93) | 93. (94) | 76.1 (96) | 82.9 (93) | 13.9 (101) |
| 46 | 1. (1) | 0.9 (14) | 11. (17) | 1.7 (11) | 2.8 (16) | 3. (16) | 0. (-1) | 5.7 (11) |
| 47 | 1. (1) | 1.5 (134) | 2.7 (104) | 8. (92) | 1.2 (71) | 12.8 (102) | 20.7 (24) | 8.2 (72) |
| 48 | 6.2 (424) | 5.2 (188) | 11.5 (160) | 1223.1 (192) | 57.1 (63) | 22.2 (215) | 84.3 (192) | 41. (195) |
| 49 | 4.1 (859) | 16.8 (733) | 172.1 (872) | 3059.3 (872) | 6.8 (451) | 54.5 (715) | 57.5 (157) | 14.1 (754) |
| 50 | 1. (1) | 1.2 (95) | 1.2 (41) | 9.5 (87) | 3. (2) | 2.7 (81) | 2.5 (2) | 4.5 (40) |
| 51 | 1. (1) | 2.2 (4) | 2.5 (49) | 13. (46) | 2.2 (49) | 6.1 (58) | 2.2 (32) | 13.6 (35) |
| 52 | 1.2 (638) | 2.7 (532) | 5.8 (39) | 34.4 (267) | 283.8 (255) | 38.5 (292) | 37.8 (281) | 23.4 (565) |

# 2 links to individual test reports

These are links to each test report. The number in square brackets to right of the link is the number of integrals in the test. The list of numbers in the curly brackets after that (if any) is the list of the integrals in that specific test which were solved by any CAS which are known not to have antiderivative. This makes it easier to find these integrals and do more investigation into them.

## 2.1 Tests completed

1. 0 Independent test suites/Apostol Problems [175]
2. 0 Independent test suites/Bondarenko Problems [35]
3. 0 Independent test suites/Bronstein Problems [14]
4. 0 Independent test suites/Charlwood Problems [50]
5. 0 Independent test suites/Hearn Problems [284] { **Mathematica: 281. Maple: 281. Maxima: 145.** }

6. 0 Independent test suites/Hebisch Problems [7]
7. 0 Independent test suites/Jeffrey Problems [9]
8. 0 Independent test suites/Moses Problems [113]
9. 0 Independent test suites/Stewart Problems [376]
10. 0 Independent test suites/Timofeev Problems [705]
11. 0 Independent test suites/Welz Problems [91]
12. 0 Independent test suites/Wester Problems [8]
13. 1 Algebraic functions/1.1 Binomial products/1.1.1Linear/1.1.1.2(a+bx)^m(c+dx)^n [1917]
14. 1 Algebraic functions/1.1 Binomial products/1.1.1Linear/1.1.1.3(a+bx)^m(c+dx)^n(e+fx)^p [3173]
15. 1 Algebraic functions/1.1 Binomial products/1.1.1Linear/1.1.1.4(a+bx)^m(c+dx)^n(e+fx)^p(g+hx)^q [158]
16. 1 Algebraic functions/1.1 Binomial products/1.1.1Linear/1.1.1.5P(x)(a+bx)^m(c+dx)^n [142]
17. 1 Algebraic functions/1.1 Binomial products/1.1.2Quadratic/1.1.2.2(cx)^m(a+bx^2)^p [1071]
18. 1 Algebraic functions/1.1 Binomial products/1.1.2Quadratic/1.1.2.3(a+bx^2)^p(c+dx^2)^q [346]
19. 1 Algebraic functions/1.1 Binomial products/1.1.2Quadratic/1.1.2.4(ex)^m(a+bx^2)^p(c+dx^2)^q [1156]
20. 1 Algebraic functions/1.1 Binomial products/1.1.2Quadratic/1.1.2.5(a+bx^2)^p(c+dx^2)^q(e+fx^2)^r [115]
21. 1 Algebraic functions/1.1 Binomial products/1.1.2Quadratic/1.1.2.6(gx)^m(a+bx^2)^p(c+dx^2)^q(e+fx^2)^r [51]
22. 1 Algebraic functions/1.1 Binomial products/1.1.2Quadratic/1.1.2.8P(x)(cx)^m(a+bx^2)^p [174]
23. 1 Algebraic functions/1.1 Binomial products/1.1.3General/1.1.3.2(cx)^m(a+bx^n)^p [3071]
24. 1 Algebraic functions/1.1 Binomial products/1.1.3General/1.1.3.3(a+bx^n)^p(c+dx^n)^q [286]
25. 1 Algebraic functions/1.1 Binomial products/1.1.3General/1.1.3.4(ex)^m(a+bx^n)^p(c+dx^n)^q [912]
26. 1 Algebraic functions/1.1 Binomial products/1.1.3General/1.1.3.6(gx)^m(a+bx^n)^p(c+dx^n)^q(e+fx^n)^r [46]
27. 1 Algebraic functions/1.1 Binomial products/1.1.3General/1.1.3.8P(x)(cx)^m(a+bx^n)^p [582]
28. 1 Algebraic functions/1.1 Binomial products/1.1.4Improper/1.1.4.2(cx)^m(ax^j+bx^n)^p [454]
29. 1 Algebraic functions/1.1 Binomial products/1.1.4Improper/1.1.4.3(ex)^m(ax^j+bx^k)^p(c+dx^n)^q [298]
30. 1 Algebraic functions/1.2 Trinomial products/1.2.1Quadratic/1.2.1.1(a+bx+cx^2)^p [143]
31. 1 Algebraic functions/1.2 Trinomial products/1.2.1Quadratic/1.2.1.2(d+ex)^m(a+bx+cx^2)^p [2580]
32. 1 Algebraic functions/1.2 Trinomial products/1.2.1Quadratic/1.2.1.3(d+ex)^m(f+gx)(a+bx+cx^2)^p [2645]
33. 1 Algebraic functions/1.2 Trinomial products/1.2.1Quadratic/1.2.1.4(d+ex)^m(f+gx)^n(a+bx+cx^2)^p [958]
34. 1 Algebraic functions/1.2 Trinomial products/1.2.1Quadratic/1.2.1.5(a+bx+cx^2)^p(d+ex+fx^2)^q [123]
35. 1 Algebraic functions/1.2 Trinomial products/1.2.1Quadratic/1.2.1.6(g+hx)^m(a+bx+cx^2)^p(d+ex+fx^2)^q [143]
36. 1 Algebraic functions/1.2 Trinomial products/1.2.1Quadratic/1.2.1.9P(x)(d+ex)^m(a+bx+cx^2)^p [400]

# 3 Listing of integrals solved by CAS which has no known antiderivatives

## 3.1 Test file Number [5] 0_Independent_test_suites/Hear

### 3.1.1 Mathematica

**Integral number [281]**

$$\int \left(\sqrt{9-4\sqrt{2}}x-\sqrt{2}\sqrt{1+4x+2x^2+x^4}\right)\,dx$$

**[B]** time = 6.07292 (sec), size = 3168 ,normalized size = 66.

Result too large to show

```
[In] Integrate[Sqrt[9 - 4*Sqrt[2]]*x - Sqrt[2]*Sqrt[1 + 4*x + 2*x^2 + x^4],x]
```

```
[Out] (Sqrt[9 - 4*Sqrt[2]]*x^2)/2 - (Sqrt[2]*x*Sqrt[1 + 4*x + 2*x^2 + x
^4])/3 - (2*Sqrt[2]*((6*(x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0
])^2*(-(EllipticF[ArcSin[Sqrt[-(((1 + x)*(Root[1 + 3*#1 - #1^2 +
#1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))/((x - Ro
ot[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*(1 + Root[1 + 3*#1 - #1^2 +
#1^3 & , 3, 0])))]], ((Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Ro
ot[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(1 + Root[1 + 3*#1 - #1^2 +
#1^3 & , 3, 0]))/((1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(Ro
ot[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3
 & , 3, 0]))]*Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0]) + EllipticPi
[(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])/(-Root[1 + 3*#1 - #1
^2 + #1^3 & , 1, 0] + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]), Arc
Sin[Sqrt[-(((1 + x)*(Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root
[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))/((x - Root[1 + 3*#1 - #1^2 +
#1^3 & , 1, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])))]],
((Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 +
#1^3 & , 2, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))/((1
+ Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(Root[1 + 3*#1 - #1^2 +
#1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))]*(1 + Ro
ot[1 + 3*#1 - #1^2 + #1^3 & , 1, 0]))*Sqrt[(x - Root[1 + 3*#1 - #
1^2 + #1^3 & , 2, 0])/((x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0]
)*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0]))]*(-1 - Root[1 + 3*
#1 - #1^2 + #1^3 & , 3, 0])*Sqrt[(x - Root[1 + 3*#1 - #1^2 + #1^3
 & , 3, 0])/((x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*(1 + Roo
t[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))]*Sqrt[-(((1 + x)*(Root[1 + 3
*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3,
0]))/((x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*(1 + Root[1 + 3
*#1 - #1^2 + #1^3 & , 3, 0])))])/(Sqrt[1 + 4*x + 2*x^2 + x^4]*(Ro
ot[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3
 & , 3, 0])) + (2*EllipticF[ArcSin[Sqrt[((1 + x)*(-Root[1 + 3*#1
- #1^2 + #1^3 & , 1, 0] + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))
/((x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*(1 + Root[1 + 3*#1
- #1^2 + #1^3 & , 3, 0]))]], ((Root[1 + 3*#1 - #1^2 + #1^3 & , 1,
 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(-1 - Root[1 + 3*#1
- #1^2 + #1^3 & , 3, 0]))/((-1 - Root[1 + 3*#1 - #1^2 + #1^3 & ,
2, 0])*(Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #
1^2 + #1^3 & , 3, 0]))]*(x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0
])^2*Sqrt[(x - Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])/((x - Root[
1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^
3 & , 2, 0]))]*(-1 - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])*Sqrt[
(x - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])/((x - Root[1 + 3*#1 -
 #1^2 + #1^3 & , 1, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0
]))]*Sqrt[((1 + x)*(-Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] + Root
[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))/((x - Root[1 + 3*#1 - #1^2 +
#1^3 & , 1, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))])/(S
qrt[1 + 4*x + 2*x^2 + x^4]*(-Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0
] + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])) + ((1 + x)*(x - Root[
1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(x - Root[1 + 3*#1 - #1^2 + #1^
3 & , 3, 0]) + (x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])^2*(1 +
 Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*Sqrt[(x - Root[1 + 3*#1 -
 #1^2 + #1^3 & , 2, 0])/((x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1,
0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0]))]*Sqrt[(x - Root[
1 + 3*#1 - #1^2 + #1^3 & , 3, 0])/((x - Root[1 + 3*#1 - #1^2 + #1
^3 & , 1, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))]*Sqrt[
```

```
-(((1 + x)*(Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1
 - #1^2 + #1^3 & , 3, 0]))/((x - Root[1 + 3*#1 - #1^2 + #1^3 & ,
1, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])))]*(1 + Root[1
 + 3*#1 - #1^2 + #1^3 & , 3, 0])*((EllipticE[ArcSin[Sqrt[-(((1 +
x)*(Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2
+ #1^3 & , 3, 0]))/((x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*(
1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])))]], ((Root[1 + 3*#1 -
 #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(
1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))/((1 + Root[1 + 3*#1 -
 #1^2 + #1^3 & , 2, 0])*(Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] -
Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))]*(1 + Root[1 + 3*#1 - #1^
2 + #1^3 & , 2, 0]))/(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])
- (EllipticPi[(1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0])/(-Root[
1 + 3*#1 - #1^2 + #1^3 & , 1, 0] + Root[1 + 3*#1 - #1^2 + #1^3 &
, 3, 0]), ArcSin[Sqrt[-(((1 + x)*(Root[1 + 3*#1 - #1^2 + #1^3 & ,
 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))/((x - Root[1 + 3
*#1 - #1^2 + #1^3 & , 1, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & ,
 3, 0])))]], ((Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3
*#1 - #1^2 + #1^3 & , 2, 0])*(1 + Root[1 + 3*#1 - #1^2 + #1^3 & ,
 3, 0]))/((1 + Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(Root[1 + 3
*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3,
0]))]*(1 - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1
- #1^2 + #1^3 & , 2, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]))
/(-Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] + Root[1 + 3*#1 - #1^2 +
 #1^3 & , 3, 0]) + (EllipticF[ArcSin[Sqrt[-(((1 + x)*(Root[1 + 3*
#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0
]))/((x - Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0])*(1 + Root[1 + 3*
#1 - #1^2 + #1^3 & , 3, 0])))]], ((Root[1 + 3*#1 - #1^2 + #1^3 &
, 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 2, 0])*(1 + Root[1 + 3*
#1 - #1^2 + #1^3 & , 3, 0]))/((1 + Root[1 + 3*#1 - #1^2 + #1^3 &
, 2, 0])*(Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] - Root[1 + 3*#1 -
 #1^2 + #1^3 & , 3, 0]))]*(Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0]
+ Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0]*(-Root[1 + 3*#1 - #1^2 +
#1^3 & , 1, 0] - Root[1 + 3*#1 - #1^2 + #1^3 & , 3, 0]) - Root[1
+ 3*#1 - #1^2 + #1^3 & , 3, 0]))/((1 + Root[1 + 3*#1 - #1^2 + #1^
3 & , 1, 0])*(-Root[1 + 3*#1 - #1^2 + #1^3 & , 1, 0] + Root[1 + 3
*#1 - #1^2 + #1^3 & , 3, 0])))))/Sqrt[1 + 4*x + 2*x^2 + x^4]))/3
```

### 3.1.2 Maple

**Integral number [281]**

$$\int \left(\sqrt{9-4\sqrt{2}}x - \sqrt{2}\sqrt{1+4x+2x^2+x^4}\right)\,dx$$

**[A]** time = 0.504 (sec), size = 4640 ,normalized size = 96.67

output too large to display

```
[In]  int(-2^(1/2)*(x^4+2*x^2+4*x+1)^(1/2)+x*(-1+2*2^(1/2)),x)
```

```
[Out] 1/2*x^2*(-1+2*2^(1/2))-2^(1/2)*(1/3*x*(x^4+2*x^2+4*x+1)^(1/2)+4/3
*(-4/3-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*
3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(
(1/2*(26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*
(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(1+x)/(1/
6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1/
2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/3
*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2)*(x+1
/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3)^2*((-1/3*
(26+6*33^(1/2))^(1/3)+8/3/(26+6*33^(1/2))^(1/3)+4/3)*(x-1/6*(26+6
*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3-1/2*I*3^(1/2)*(-1/
3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(1/6*(26+6*33
^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3+1/2*I*3^(1/2)*(-1/3*(
26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26+6*33^
(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2)*((-1/3*(26+6*3
3^(1/2))^(1/3)+8/3/(26+6*33^(1/2))^(1/3)+4/3)*(x-1/6*(26+6*33^(1/
2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3+1/2*I*3^(1/2)*(-1/3*(26+6
*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(1/6*(26+6*33^(1/2))
^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1/2)*(-1/3*(26+6*33
^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26+6*33^(1/2))^
(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2)/(1/2*(26+6*33^(1/2))^
(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))
^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-1/3*(26+6*33^(1/2))^(1/3)+8/
3/(26+6*33^(1/2))^(1/3)+4/3)/((1+x)*(x+1/3*(26+6*33^(1/2))^(1/3)-
8/3/(26+6*33^(1/2))^(1/3)-1/3)*(x-1/6*(26+6*33^(1/2))^(1/3)+4/3/(
26+6*33^(1/2))^(1/3)-1/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3
)-8/3/(26+6*33^(1/2))^(1/3)))*(x-1/6*(26+6*33^(1/2))^(1/3)+4/3/(2
6+6*33^(1/2))^(1/3)-1/3+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)
-8/3/(26+6*33^(1/2))^(1/3))))^(1/2)*EllipticF(((1/2*(26+6*33^(1/2
))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/
2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(1+x)/(1/6*(26+6*33^(1/2))^
(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^
(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26+6*33^(1/2))^(
1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2),((-1/2*(26+6*33^(1/2))
^(1/3)+4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2)
)^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(-4/3-1/6*(26+6*33^(1/2))^(1/
3)+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^
(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-4/3-1/6*(26+6*33^(1/2))^(1/3)
+4/3/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1
/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-1/2*(26+6*33^(1/2))^(1/3)+4/(26
+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/
(26+6*33^(1/2))^(1/3))))^(1/2))+4*(-4/3-1/6*(26+6*33^(1/2))^(1/3)
+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1
/3)-8/3/(26+6*33^(1/2))^(1/3)))*((1/2*(26+6*33^(1/2))^(1/3)-4/(26
+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/
(26+6*33^(1/2))^(1/3)))*(1+x)/(1/6*(26+6*33^(1/2))^(1/3)-4/3/(26+
6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8
/3/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6
*33^(1/2))^(1/3)-1/3))^(1/2)*(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26
+6*33^(1/2))^(1/3)-1/3)^2*((-1/3*(26+6*33^(1/2))^(1/3)+8/3/(26+6*
33^(1/2))^(1/3)+4/3)*(x-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1
/2))^(1/3)-1/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+
6*33^(1/2))^(1/3)))/(1/6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2)
)^(1/3)+4/3+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*3
```

```
3^(1/2))^(1/3)))/(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))
^(1/3)-1/3))^(1/2)*((-1/3*(26+6*33^(1/2))^(1/3)+8/3/(26+6*33^(1/2
))^(1/3)+4/3)*(x-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1
/3)-1/3+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1
/2))^(1/3)))/(1/6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)
+4/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2)
)^(1/3)))/(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-
1/3))^(1/2)/(1/2*(26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/
2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)
))/(-1/3*(26+6*33^(1/2))^(1/3)+8/3/(26+6*33^(1/2))^(1/3)+4/3)/((1
+x)*(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3)*(
x-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3-1/2*I*3
^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(x
-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3+1/2*I*3^
(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3))))^(1
/2)*((-1/3*(26+6*33^(1/2))^(1/3)+8/3/(26+6*33^(1/2))^(1/3)+1/3)*E
llipticF(((1/2*(26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*
I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))
*(1+x)/(1/6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1
/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3
)))/(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^
(1/2),((-1/2*(26+6*33^(1/2))^(1/3)+4/(26+6*33^(1/2))^(1/3)-1/2*I*
3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(
-4/3-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*3^
(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-4
/3-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1
/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-1/2
*(26+6*33^(1/2))^(1/3)+4/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/
3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3))))^(1/2))+(-4/3
+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3))*EllipticPi(
((1/2*(26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)
*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(1+x)/(1
/6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1
/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/
3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2),(1/
6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1/
2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(1/2*(
26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*
(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3))),((-1/2*(26+6*33
^(1/2))^(1/3)+4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*3
3^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(-4/3-1/6*(26+6*33^(1/
2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26+6*33^
(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-4/3-1/6*(26+6*33^(1/2)
)^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1
/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-1/2*(26+6*33^(1/2))^(1/3
)+4/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/
3)-8/3/(26+6*33^(1/2))^(1/3))))^(1/2)))+2/3*((1+x)*(x-1/6*(26+6*3
3^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3-1/2*I*3^(1/2)*(-1/3*
(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(x-1/6*(26+6*33
^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3+1/2*I*3^(1/2)*(-1/3*(
26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))+(-4/3-1/6*(26+6*
33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26
+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*((1/2*(26+6*33^(1/
2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1
```

```
/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(1+x)/(1/6*(26+6*33^(1/2))
^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1/2)*(-1/3*(26+6*33
^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26+6*33^(1/2))^
(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2)*(x+1/3*(26+6*33^(1/2)
)^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3)^2*((-1/3*(26+6*33^(1/2))^(
1/3)+8/3/(26+6*33^(1/2))^(1/3)+4/3)*(x-1/6*(26+6*33^(1/2))^(1/3)+
4/3/(26+6*33^(1/2))^(1/3)-1/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))
^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(1/6*(26+6*33^(1/2))^(1/3)-4/3
/(26+6*33^(1/2))^(1/3)+4/3+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1
/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/
(26+6*33^(1/2))^(1/3)-1/3))^(1/2)*((-1/3*(26+6*33^(1/2))^(1/3)+8/
3/(26+6*33^(1/2))^(1/3)+4/3)*(x-1/6*(26+6*33^(1/2))^(1/3)+4/3/(26
+6*33^(1/2))^(1/3)-1/3+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-
8/3/(26+6*33^(1/2))^(1/3)))/(1/6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*
33^(1/2))^(1/3)+4/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3
/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*3
3^(1/2))^(1/3)-1/3))^(1/2)*((1/2*(26+6*33^(1/2))^(1/3)-4/(26+6*33
^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6
*33^(1/2))^(1/3))+(1/6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^
(1/3)+1/3-1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^
(1/2))^(1/3)))*(-1/3*(26+6*33^(1/2))^(1/3)+8/3/(26+6*33^(1/2))^(1
/3)+1/3)+(-1/3*(26+6*33^(1/2))^(1/3)+8/3/(26+6*33^(1/2))^(1/3)+1/
3)^2)/(1/2*(26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^
(1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-1
/3*(26+6*33^(1/2))^(1/3)+8/3/(26+6*33^(1/2))^(1/3)+4/3)*EllipticF
(((1/2*(26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2
)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(1+x)/(
1/6*(26+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(
1/2)*(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1
/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2),((
-1/2*(26+6*33^(1/2))^(1/3)+4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*
(-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(-4/3-1/6
*(26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-
1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-4/3-1/6*(
26+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/
3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-1/2*(26+6*3
3^(1/2))^(1/3)+4/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26+6*
33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3))))^(1/2))+(-4/3-1/6*(26
+6*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*
(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*EllipticE(((1/2
*(26+6*33^(1/2))^(1/3)-4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/
3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(1+x)/(1/6*(2
6+6*33^(1/2))^(1/3)-4/3/(26+6*33^(1/2))^(1/3)+4/3-1/2*I*3^(1/2)*(
-1/3*(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(x+1/3*(26
+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3))^(1/2),((-1/2*(
26+6*33^(1/2))^(1/3)+4/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*
(26+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(-4/3-1/6*(26+6
*33^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(2
6+6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-4/3-1/6*(26+6*3
3^(1/2))^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/2*I*3^(1/2)*(-1/3*(26+
6*33^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))/(-1/2*(26+6*33^(1/2
))^(1/3)+4/(26+6*33^(1/2))^(1/3)+1/2*I*3^(1/2)*(-1/3*(26+6*33^(1/
2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3))))^(1/2))/(-1/3*(26+6*33^(1/2
))^(1/3)+8/3/(26+6*33^(1/2))^(1/3)+4/3)))/((1+x)*(x+1/3*(26+6*33^
```

```
(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)-1/3)*(x-1/6*(26+6*33^(1/2)
)^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3-1/2*I*3^(1/2)*(-1/3*(26+6*3
3^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3)))*(x-1/6*(26+6*33^(1/2))
^(1/3)+4/3/(26+6*33^(1/2))^(1/3)-1/3+1/2*I*3^(1/2)*(-1/3*(26+6*33
^(1/2))^(1/3)-8/3/(26+6*33^(1/2))^(1/3))))^(1/2))
```

### 3.1.3 Maxima

**Integral number [145]**

$$\int x \cos(k \csc(x)) \cot(x) \csc(x)\, dx$$

**[A]** time = 0.132088 (sec), size = 324 ,normalized size = 23.14

$$-\frac{\left(xe^{\left(\frac{4k\cos(2x)\cos(x)}{\cos(2x)^2+\sin(2x)^2-2\cos(2x)+1}+\frac{4k\sin(2x)\sin(x)}{\cos(2x)^2+\sin(2x)^2-2\cos(2x)+1}\right)}+xe^{\left(\frac{4k\cos(x)}{\cos(2x)^2+\sin(2x)^2-2\cos(2x)+1}\right)}\right)e^{\left(-\frac{2k\cos(2x)\cos(x)}{\cos(2x)^2+\sin(2x)^2-2\cos(2x)+1}-\frac{2k\sin(2x)\sin(x)}{\cos(2x)^2+\sin(2x)^2-2\cos(2x)+1}-}}{2k}$$

```
[In]  integrate(x*cos(x)*cos(k/sin(x))/sin(x)^2,x, algorithm="maxima")
```

```
[Out] -1/2*(x*e^(4*k*cos(2*x)*cos(x)/(cos(2*x)^2 + sin(2*x)^2 - 2*cos(2
*x) + 1) + 4*k*sin(2*x)*sin(x)/(cos(2*x)^2 + sin(2*x)^2 - 2*cos(2
*x) + 1)) + x*e^(4*k*cos(x)/(cos(2*x)^2 + sin(2*x)^2 - 2*cos(2*x)
 + 1)))*e^(-2*k*cos(2*x)*cos(x)/(cos(2*x)^2 + sin(2*x)^2 - 2*cos(
2*x) + 1) - 2*k*sin(2*x)*sin(x)/(cos(2*x)^2 + sin(2*x)^2 - 2*cos(
2*x) + 1) - 2*k*cos(x)/(cos(2*x)^2 + sin(2*x)^2 - 2*cos(2*x) + 1)
)*sin(2*(k*cos(x)*sin(2*x) - k*cos(2*x)*sin(x) + k*sin(x))/(cos(2
*x)^2 + sin(2*x)^2 - 2*cos(2*x) + 1))/k
```

# 4 Listing of grading functions

The following are the current version of the grading functions used for grading the quality of the antiderivative with reference to the optimal antiderivative included in the test suite.

There is a version for Maple and another one for Mathematica/Rubi. The following are links to the source code.

The following are the listings of source code of the grading functions.

Mathematica and Rubi grading function

```
(* Original version thanks to Albert Rich emailed on 03/21/2017 *)
(* ::Package:: *)

(* ::Subsection:: *)
(*GradeAntiderivative[result,optimal]*)

```

```
7
8  (* ::Text:: *)
9  (*If result and optimal are mathematical expressions, *)
10 (*          GradeAntiderivative[result,optimal] returns*)
11 (*  "F" if the result fails to integrate an expression that*)
12 (*      is integrable*)
13 (*  "C" if result involves higher level functions than necessary*)
14 (*  "B" if result is more than twice the size of the optimal*)
15 (*      antiderivative*)
16 (*  "A" if result can be considered optimal*)
17
18
19 GradeAntiderivative[result_,optimal_] :=
20   If[ExpnType[result]<=ExpnType[optimal],
21     If[FreeQ[result,Complex] || Not[FreeQ[optimal,Complex]],
22       If[LeafCount[result]<=2*LeafCount[optimal],
23         "A",
24       "B"],
25     "C"],
26   If[FreeQ[result,Integrate] && FreeQ[result,Int],
27     "C",
28   "F"]]
29
30
31 (* ::Text:: *)
32 (*The following summarizes the type number assigned an *)
33 (*expression based on the functions it involves*)
34 (*1 = rational function*)
35 (*2 = algebraic function*)
36 (*3 = elementary function*)
37 (*4 = special function*)
38 (*5 = hyperpergeometric function*)
39 (*6 = appell function*)
40 (*7 = rootsum function*)
41 (*8 = integrate function*)
42 (*9 = unknown function*)
43
44
45 ExpnType[expn_] :=
46   If[AtomQ[expn],
47     1,
48   If[ListQ[expn],
49     Max[Map[ExpnType,expn]],
50   If[Head[expn]===Power,
51     If[IntegerQ[expn[[2]]],
52       ExpnType[expn[[1]]],
53     If[Head[expn[[2]]]===Rational,
54       If[IntegerQ[expn[[1]]] || Head[expn[[1]]]===Rational,
55         1,
56       Max[ExpnType[expn[[1]]],2]],
57     Max[ExpnType[expn[[1]]],ExpnType[expn[[2]]],3]]],
58   If[Head[expn]===Plus || Head[expn]===Times,
59     Max[ExpnType[First[expn]],ExpnType[Rest[expn]]],
60   If[ElementaryFunctionQ[Head[expn]],
61     Max[3,ExpnType[expn[[1]]]],
62   If[SpecialFunctionQ[Head[expn]],
63     Apply[Max,Append[Map[ExpnType,Apply[List,expn]],4]],
64   If[HypergeometricFunctionQ[Head[expn]],
65     Apply[Max,Append[Map[ExpnType,Apply[List,expn]],5]],
```

```
  If[AppellFunctionQ[Head[expn]],
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],6]],
  If[Head[expn]===RootSum,
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],7]],
  If[Head[expn]===Integrate || Head[expn]===Int,
    Apply[Max,Append[Map[ExpnType,Apply[List,expn]],8]],
  9]]]]]]]]]]


ElementaryFunctionQ[func_] :=
  MemberQ[{
        Exp,Log,
        Sin,Cos,Tan,Cot,Sec,Csc,
        ArcSin,ArcCos,ArcTan,ArcCot,ArcSec,ArcCsc,
        Sinh,Cosh,Tanh,Coth,Sech,Csch,
        ArcSinh,ArcCosh,ArcTanh,ArcCoth,ArcSech,ArcCsch
},func]


SpecialFunctionQ[func_] :=
  MemberQ[{
        Erf, Erfc, Erfi,
        FresnelS, FresnelC,
        ExpIntegralE, ExpIntegralEi, LogIntegral,
        SinIntegral, CosIntegral, SinhIntegral, CoshIntegral,
        Gamma, LogGamma, PolyGamma,
        Zeta, PolyLog, ProductLog,
        EllipticF, EllipticE, EllipticPi
},func]


HypergeometricFunctionQ[func_] :=
  MemberQ[{Hypergeometric1F1,Hypergeometric2F1,HypergeometricPFQ},
    func]


AppellFunctionQ[func_] :=
  MemberQ[{AppellF1},func]
```

Maple grading function

```
# File: GradeAntiderivative.mpl
# Original version thanks to Albert Rich emailed on 03/21/2017

#Nasser 03/22/2017  Use Maple leaf count instead since buildin
#Nasser 03/23/2017  missing 'ln' for ElementaryFunctionQ added
#Nasser 03/24/2017  corrected the check for complex result
#Nasser 10/27/2017  check for leafsize and do not call ExpnType()
#                   if leaf size is "too large". Set at 500,000

GradeAntiderivative := proc(result,optimal)
local leaf_count_result, leaf_count_optimal,ExpnType_result,ExpnType_optimal;

      leaf_count_result:=leafcount(result);
      #do NOT call ExpnType() if leaf size is too large. Recursion problem
      if leaf_count_result > 500000 then
          return "B";
      fi;

```

```
      leaf_count_optimal:=leafcount(optimal);

      ExpnType_result:=ExpnType(result);
      ExpnType_optimal:=ExpnType(optimal);

# If result and optimal are mathematical expressions,
#  GradeAntiderivative[result,optimal] returns
#   "F" if the result fails to integrate an expression that
#       is integrable
#   "C" if result involves higher level functions than necessary
#   "B" if result is more than twice the size of the optimal
#       antiderivative
#   "A" if result can be considered optimal

  #This check below actually is not needed, since I only
  #call this grading only for passed integrals. i.e. I check
  #for "F" before calling this. But no harm of keeping it here.
  #just in case.


  if not type(result,freeof('int')) then
     return "F";
  end if;


  if ExpnType_result<=ExpnType_optimal then
      if is_contains_complex(result) then
         if is_contains_complex(optimal) then
            #both result and optimal complex
            if leaf_count_result<=2*leaf_count_optimal then
               return "A";
            else
               return "B";
            end if
         else #result contains complex but optimal is not
            return "C";
         end if
      else # result do not contain complex
           # this assumes optimal do not as well
         if leaf_count_result<=2*leaf_count_optimal then
            return "A";
         else
            return "B";
         end if
      end if
  else #ExpnType(result) > ExpnType(optimal)
      return "C";
  end if

end proc:

#
# is_contains_complex(result)
# takes expressions and returns true if it contains "I" else false
#
#Nasser 032417
is_contains_complex:= proc(expression)
  return (has(expression,I));
end proc:
```

```
78
79  # The following summarizes the type number assigned an expression
80  # based on the functions it involves
81  # 1 = rational function
82  # 2 = algebraic function
83  # 3 = elementary function
84  # 4 = special function
85  # 5 = hyperpergeometric function
86  # 6 = appell function
87  # 7 = rootsum function
88  # 8 = integrate function
89  # 9 = unknown function
90
91  ExpnType := proc(expn)
92    if type(expn,'atomic') then
93      1
94    elif type(expn,'list') then
95      apply(max,map(ExpnType,expn))
96    elif type(expn,'sqrt') then
97      if type(op(1,expn),'rational') then
98        1 else
99      max(2,ExpnType(op(1,expn)))
100     end if
101   elif type(expn,'`^`') then
102     if type(op(2,expn),'integer') then
103       ExpnType(op(1,expn))
104     elif type(op(2,expn),'rational') then
105       if type(op(1,expn),'rational') then
106         1 else
107       max(2,ExpnType(op(1,expn))) end if else
108     max(3,ExpnType(op(1,expn)),ExpnType(op(2,expn)))
109     end if
110   elif type(expn,'`+`') or type(expn,'`*`') then
111     max(ExpnType(op(1,expn)),max(ExpnType(rest(expn))))
112   elif ElementaryFunctionQ(op(0,expn)) then
113     max(3,ExpnType(op(1,expn)))
114   elif SpecialFunctionQ(op(0,expn)) then
115     max(4,apply(max,map(ExpnType,[op(expn)])))
116   elif HypergeometricFunctionQ(op(0,expn)) then
117     max(5,apply(max,map(ExpnType,[op(expn)])))
118   elif AppellFunctionQ(op(0,expn)) then
119     max(6,apply(max,map(ExpnType,[op(expn)])))
120   elif op(0,expn)='int' or op(0,expn)='integrate' then
121     max(8,apply(max,map(ExpnType,[op(expn)]))) else
122   9
123   end if
124 end proc:
125
126
127 ElementaryFunctionQ := proc(func)
128   member(func,[
129         exp,log,ln,
130         sin,cos,tan,cot,sec,csc,
131         arcsin,arccos,arctan,arccot,arcsec,arccsc,
132         sinh,cosh,tanh,coth,sech,csch,
133         arcsinh,arccosh,arctanh,arccoth,arcsech,arccsch])
134 end proc:
135
136 SpecialFunctionQ := proc(func)
```

```
  member(func,[
        erf,erfc,erfi,
        FresnelS,FresnelC,
        Ei,Ei,Li,Si,Ci,Shi,Chi,
        GAMMA,lnGAMMA,Psi,Zeta,polylog,LambertW,
        EllipticF,EllipticE,EllipticPi])
end proc:

HypergeometricFunctionQ := proc(func)
  member(func,[Hypergeometric1F1,hypergeom,HypergeometricPFQ])
end proc:

AppellFunctionQ := proc(func)
  member(func,[AppellF1])
end proc:

# u is a sum or product.  rest(u) returns all but the
# first term or factor of u.
rest := proc(u) local v;
  if nops(u)=2 then
    op(2,u) else
  apply(op(0,u),op(2..nops(u),u))
  end if
end proc:

#leafcount(u) returns the number of nodes in u.
#Nasser 3/23/17  Replaced by build-in leafCount from package in Maple
leafcount := proc(u)
   MmaTranslator[Mma][LeafCount](u);
end proc:
```