FFT Library

Yuzhi Zhao

October 25, 2017

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
CA	Commonality Analysis
IM	Instance Module
MSE	Mean Squared Error
o_i	Output Data
e_i	Expected Output Data

Contents

List of Tables

List of Figures

3 General Information

The following section provides an overview of the Verification and Validation (V & V) Plan for a FFT library.

3.1 Purpose

The main purpose of this document is to describe the verification and validation process that will be used to test a FFT Library.This document is intended to be used as a reference for all future testing and will be used to increase confidence in the software implementation.
This document will be used as a starting point for the verification and validation report. The test cases presented within this document will be executed and the output will be analyzed to determine if the library is implemented correctly.

3.2 Scope

The whole library includes four FFT or IFFT calculation functions. All tests should be applied based on this scope.

3.3 Overview of Document

The following sections provides more details about the V&V of a FFT Library. Information about verification tools, automated testing approaches will be stated. And test cases for all system testing and part of unit testing will be provided.

4 Plan

4.1 Software Description

The software being tested is a library for FFT algorithm. Users choose different FFT or IFFT functions and give proper input datas to complete a FFT or IFFT calculation. The library includes radix-2 and radix-3 FFT(and IFFT) calculation functions.

4.2 Test Team

Yuzhi Zhao

4.3 Automated Testing Approach

A unit testing framework will be implemented in both unit testing and system testing.
Script will be used to call all the test cases in test suite.
Test coverage analysis will be applied to measure code coverage.
Compiler can do syntax check automatically.

4.4 Verification Tools

1. Cutest as unit testing framework
2. Make as script to call test cases and execute test
3. Xcover as coverage analysis tool

4.5 Non-Testing Based Verification

Symbolic Execution

Because FFT library is based on a mathematical expression. Using Symbolic Execution can trace the path and the result can be compared with mathematical expression directly.

5 System Test Description

5.1 Tests for Functional Requirements

5.1.1 Calculation Test

Radix-2 Complex Number Calculation Function

1. **T-1:Radix-2 Complex Number FFT Calculation Function**

   Type: Functional, Dynamic, Automated

   Initial State: None

Input:
input.txt : Includes all the input datas. Two exmples of input.txt is shown in Figure 1 and Figure 2. The floating numbers can be generated by random number generator online. The source can be reached using `http://www.meridianoutpost.com/resources/etools/calculators/generator-random-real.php?` The integer numbers can be generated using `https://andrew.hedges.name/experiments/random/`.

expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab FFT library. Then expectedOutput.txts are shown in Figure 3 and Figure 4.
If the numbers of data can not satisfy 2^n, program will automatically fill with 0.

Output:
output.txt : Includes the output datas using the input data computed by this FFT library.
TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the Mean Squared Error wiil be used as the algorithm. The equation is provided below:

$$MSE = \frac{1}{n} \sum_{i=0}^{n-1} (o_i - e_i)^2 \tag{1}$$

e means expected output. o means this library's output.
If the value of MSE is below 1% of average of input datas, then this library passed the test.

How test will be performed:
Automated.
For validation purpose, datas should also be compared with results from normal DFT calculations as well. Do the same test as above but fill the output.txt with results from using DFT library.

2. **T-2:Radix-2 Complex Number IFFT Calculation Function**

   Type: Functional, Dynamic, Automated

Initial State: None

Input:
input.txt : Includes all the input datas. The datas of input testing file can use the same datas from output.txt from T- 1 shown in Figure 3 and Figure 4.

expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab IFFT library.
If the numbers of data can not satisfy 2^n, program will automatically fill with 0.

Output:
output.txt : Includes the output datas using the input data computed by this IFFT library.
TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.
How test will be performed:
Same as above.

Radix-2 Real Number Calculation Function

1. **T-3:Radix-2 Real Number FFT Calculation Function**

   Type: Functional, Dynamic, Automated

   Initial State: None

   Input:
   input.txt : Includes all the input datas. Two exmples of input.txt is shown in Figure 5 and Figure 6. The floating numbers and he integer numbers can be generated by on line random number generators.

   expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab FFT library. Then expectedOutput.txts are shown in Figure 7 and Figure 8.
   If the numbers of data can not satisfy 2^n, program will automatically

fill with 0.

Output:
output.txt : Includes the output datas using the input data computed by this FFT library.
TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:
Same as above.

2. **T-4:Radix-2 Real Number IFFT Calculation Function**

   Type: Functional, Dynamic, Automated

   Initial State: None

   Input:
   input.txt : Includes all the input datas. The datas of input testing file can use the same datas from output.txt from T-3 showed in Figure 7 and Figure 8.

   expectedOutput.txt: Includes the output datas using the same input datas but computed by Matlab IFFT library.
   If the numbers of data can not satisfy 2^n, program will automatically fill with 0.

   Output:
   output.txt : Includes the output datas using the input data computed by this IFFT library.
   TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

   How test will be performed:
   Same as above.

Radix-3 Complex Number Calculation Function

1. **T-5:Radix-3 Complex Number FFT Calculation Function**

   Type: Functional, Dynamic, Automated

   Initial State: None

   Input:
   input.txt: Same as input.txt in T-1. Reference Figure 1 and Figure 2.
   expectedOutput.txt: Same as expectedOutput.txt in T-1. Reference Figure 3 and Figure 4.
   If the numbers of data can not satisfy 3^n, program will automatically fill with 0.

   Output:
   output.txt : Includes the output datas using the input data computed by this FFT library.
   TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

   How test will be performed:
   Same as above.

2. **T-6:Radix-3 Cpmplex Number IFFT Calculation Function**

   Type: Functional, Dynamic, Automated

   Initial State: None

   Input:
   input.txt: Same as input.txt in T-2. Reference Figure 3 and Figure 4.
   expectedOutput.txt: Same as expectedOutput.txt in T-2.
   If the numbers of data can not satisfy 3^n, program will automatically fill with 0.

   Output:
   output.txt : Includes the output datas using the input data computed by this FFT library.
   TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:
Same as above.

Radix-3 Real Number Calculation Function

1. **T-7:Radix-3 Real Number FFT Calculation Function**

   Type: Functional, Dynamic, Automated

   Initial State: None

   Input:
   input.txt: Same as input.txt in T-3. Reference Figure 5 and Figure 6.
   expectedOutput.txt: Same as expectedOutput.txt in T-3. Reference Figure 7 and Figure 8.
   If the numbers of data can not satisfy 3^n, program will automatically fill with 0.

   Output:
   output.txt : Includes the output datas using the input data computed by this FFT library.
   TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

   How test will be performed:

2. **T-8:Radix-3 Real Number IFFT Calculation Function**

   Type: Functional, Dynamic, Automated

   Initial State: None

   Input:
   input.txt: Same as input.txt in T-4. Reference Figure 7 and Figure 8.
   expectedOutput.txt: Same as expectedOutput.txt in T-4.
   If the numbers of data can not satisfy 3^n, program will automatically fill with 0.

Output:
output.txt : Includes the output datas using the input data computed by this FFT library.
TestResult: pass or not pass. Whether the program passed the test is measured by an admissible error and the algorithm is same as it in T-1.

How test will be performed:
Same as above.

5.2 Tests for Nonfunctional Requirements

5.2.1 Speed Comparation Test

1. **T-9:Compare Calculation Speed with DFT calculation**

   Type: Dynamic, automated, Manual

   Initial State: None

   Input: intput.txt

   Output: Time

   How test will be performed:
   Manually compare the time with the time using DFT Library.

5.2.2 Loading Library Test

1. **T-10:Under Win X86 plateform**

   Type: Functional, Dynamic, Manual

   Initial State: None

   Input: input.txt(can be chosen from any input.txt above mentioned and call the corresponding function.) to an C Language compiler.

   Output: output.txt

   How test will be performed: Manual

2. **T-11:Under Mac OS plateform**

   Type: Functional, Dynamic, Manual

   Initial State: None

Input: input.txt(can be chosen from any input.txt above mentioned and call the corresponding function.) to an C Language compiler.

Output: output.txt

How test will be performed: Manual

3. **T-12:Different Compilers Under The Same Plateform**

   Type: Functional, Dynamic, Manual

   Initial State: None

   Input: input.txt(can be chosen from any input.txt above mentioned and call the corresponding function.) to different compilers including different versions and different languages.

   Output: output.txt

   How test will be performed: Manual

5.3 Traceability Between Test Cases and Requirements

Since CA does not include requirements part, the Test Cases will be relevanted to IM.
T-1, T-2, T-3, T-4 all relevant to IM1 in CA.
T-5, T-6, T-7, T-8 all relevant to IM2 in CA.

6 Unit Testing Plan

6.1 Input Check Test

1. **T-13: Check Numbers Of Input Data**

   Type: Functional, Dynamic

   Initial State: None

   Input: List of Input Datas

   Output: Check whether List.length equals 2^n or 3^n according to Radix

   How test will be performed: Automated Unit Test

2. **T-14: Fill Input With 0**

   Type: Functional, Dynamic

   Initial State: None

   Input: List of Input Datas

   Output: List (The List.length must equal to 2^n or 3^n according to Radix)

   How test will be performed: Automated Unit Test

3. **T-15: Check whether the input data type is the required data type corresponding to the called Function**

   Type: Functional, Dynamic

   Initial State: None

   Input: List of Input Datas

   Output: Ture or False (True means the data type is right, False means the data type is wrong.)

   How test will be performed: Automated Unit Test

6.2 Calculation Test

1. **T-16: Complex Number Multiplication Check**

   Type: Functional, Dynamic

   Initial State: None

   Input: (1 + 2i)*(2 + 3i)

   Output: (-5 + 7i)

   How test will be performed: Automated Unit Test

2. **T-17: complex number Addition Check**

   Type: Functional, Dynamic

   Initial State: None

   Input:(1 + 2i) + (2 + 3i)

   Output: (3 + 5i)

   How test will be performed: Automated Unit Test

3. **T-18:** $e^{\frac{-2\pi ki}{N}}$ **Calculation Check**

   Type: Functional, Dynamic

   Initial State: None

   Input: k = 1, N =4

   Output: -i

   How test will be performed: Automated Unit Test

6.3 Partition Test

1. **T-19:Radix-2 Partition check**

   Type: Functional, Dynamic

   Initial State: None

   Input: [0, 1, 2, 3, 4, 5, 6, 7] means [x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7]

   Output: [0, 4, 2, 6, 1, 5, 3,7] means [x_0, x_4, x_2, x_6, x_1, x_5, x_3, x_7]

   How test will be performed: Automated Unit Test

2. **T-20: Radix-2 Partition check**

   Type: Functional, Dynamic

   Initial State: None

   Input: [0, 1, 2, 3, 4, 5, 6, 7, 8] means [x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]

   Output: [0, 3, 6, 1, 4, 7, 2, 5, 8] means [x_0, x_3, x_6, x_1, x_4, x_7, x_2, x_5, x_8]

   How test will be performed: Automated Unit Test

7 Appendix

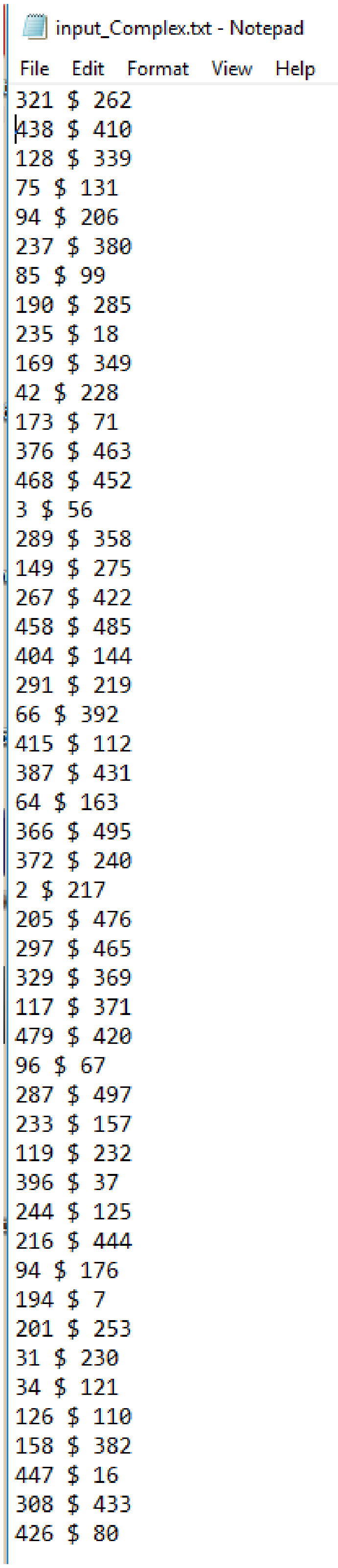
This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.

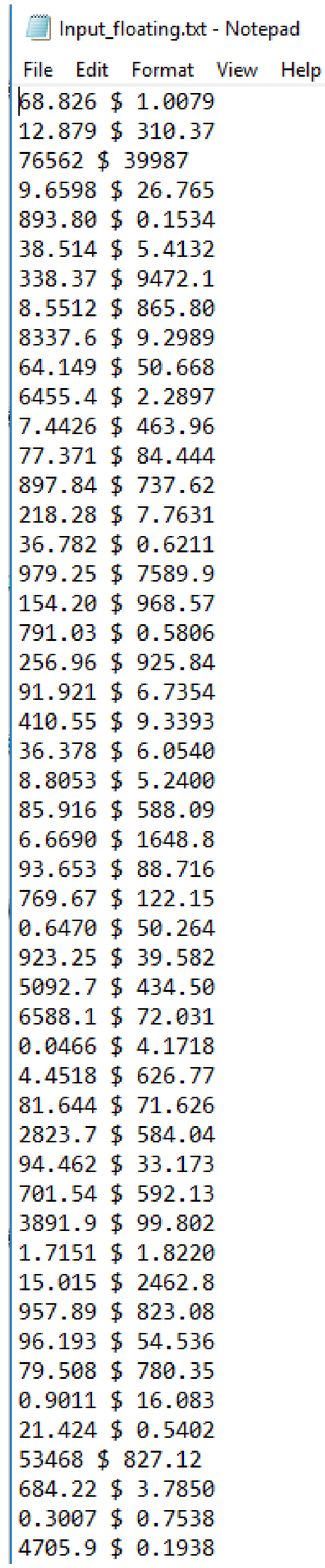


input_Complex.txt - Notepad

File Edit Format View Help

```
321 $ 262
438 $ 410
128 $ 339
75 $ 131
94 $ 206
237 $ 380
85 $ 99
190 $ 285
235 $ 18
169 $ 349
42 $ 228
173 $ 71
376 $ 463
468 $ 452
3 $ 56
289 $ 358
149 $ 275
267 $ 422
458 $ 485
404 $ 144
291 $ 219
66 $ 392
415 $ 112
387 $ 431
64 $ 163
366 $ 495
372 $ 240
2 $ 217
205 $ 476
297 $ 465
329 $ 369
117 $ 371
479 $ 420
96 $ 67
287 $ 497
233 $ 157
119 $ 232
396 $ 37
244 $ 125
216 $ 444
94 $ 176
194 $ 7
201 $ 253
31 $ 230
34 $ 121
126 $ 110
158 $ 382
447 $ 16
308 $ 433
426 $ 80
```

Figure 1: Complex Integer Numbers As Input



Input_floating.txt - Notepad

File Edit Format View Help

```
68.826 $ 1.0079
12.879 $ 310.37
76562 $ 39987
9.6598 $ 26.765
893.80 $ 0.1534
38.514 $ 5.4132
338.37 $ 9472.1
8.5512 $ 865.80
8337.6 $ 9.2989
64.149 $ 50.668
6455.4 $ 2.2897
7.4426 $ 463.96
77.371 $ 84.444
897.84 $ 737.62
218.28 $ 7.7631
36.782 $ 0.6211
979.25 $ 7589.9
154.20 $ 968.57
791.03 $ 0.5806
256.96 $ 925.84
91.921 $ 6.7354
410.55 $ 9.3393
36.378 $ 6.0540
8.8053 $ 5.2400
85.916 $ 588.09
6.6690 $ 1648.8
93.653 $ 88.716
769.67 $ 122.15
0.6470 $ 50.264
923.25 $ 39.582
5092.7 $ 434.50
6588.1 $ 72.031
0.0466 $ 4.1718
4.4518 $ 626.77
81.644 $ 71.626
2823.7 $ 584.04
94.462 $ 33.173
701.54 $ 592.13
3891.9 $ 99.802
1.7151 $ 1.8220
15.015 $ 2462.8
957.89 $ 823.08
96.193 $ 54.536
79.508 $ 780.35
0.9011 $ 16.083
21.424 $ 0.5402
53468 $ 827.12
684.22 $ 3.7850
0.3007 $ 0.7538
4705.9 $ 0.1938
```

Figure 2: Complex Floating Numbers As Input

int_complexout.txt - Notepad

File Edit Format View Help

```
11601.0000000000 & 13170.0000000000
-1302.36546460335 $ -1225.18412384423
109.886649455757 $ 255.330675629305
2272.91391891299 $ 846.985943062365
131.976286674319 $-702.095204215212
836.768977430609 $ -248.898725236191
175.989269103497 $ -694.781581201605
-394.293978330056 $ -917.236615618077
1486.24092585602 $ -413.427827584198
-1275.19352333377 $ 1492.39974564758
-642.133610919904 $ 36.5285041939520
604.375136295333 $ -566.743704316595
-384.504286488376 $ 1182.72591073215
-1154.59305847736 $ -421.887293951124
323.036400133849 $ 1554.28500178386
-461.606967075186 $-1888.40395205162
959.082172493056 $ 1003.93666717407
-882.438427727607 $ -1329.23228103844
-2023.66860739028 $ -733.727391654711
1751.19423074555 $ -566.994816780796
629.251293298174 $ -68.9349197472861
-2259.93704296279 $ -71.9229671118823
-1083.10895941491 $ 1816.56855514635
174.498757084428 $ -626.368900991037
-2590.62723128634 $1322.12921001115
-619.000000000000 $ 128.000000000000
381.198251825687 $ 109.732363768243
302.698360614686 $ 501.783098426547
1057.58922044532 $ 1527.68344300514
-593.518778679489 $ -513.422626990863
1115.37515134291 $ 825.092787157243
-1149.96102909039 $ 578.599800551132
1211.43831036296 $ -1784.32728243514
665.066313612006 $ -177.279294016356
576.643101696344 $ -1096.02120318646
-1704.30506239337 $ -241.804165345742
-816.868163747058 $ -1280.94723514531
1777.87280400399 $ -33.3569191239944
2183.51191266229 $ -159.843310385875
104.190887546521 $ -614.156140490604
-1063.49283372118 $ -192.686371603909
691.141118956884 $ -214.109016987594
879.122230681935 $ 654.003719857377
-51.4818598437252 $ 149.352228419853
91.4326924684502 $ 717.504216835858
1078.14305203795 $ 1021.10684263355
1718.83198891961 $ 754.920743119239
756.736443981558 $ 878.085553073900
1147.79783554787 $ 1121.35052874577
-291.904808705419 $ -1764.31166791979
```

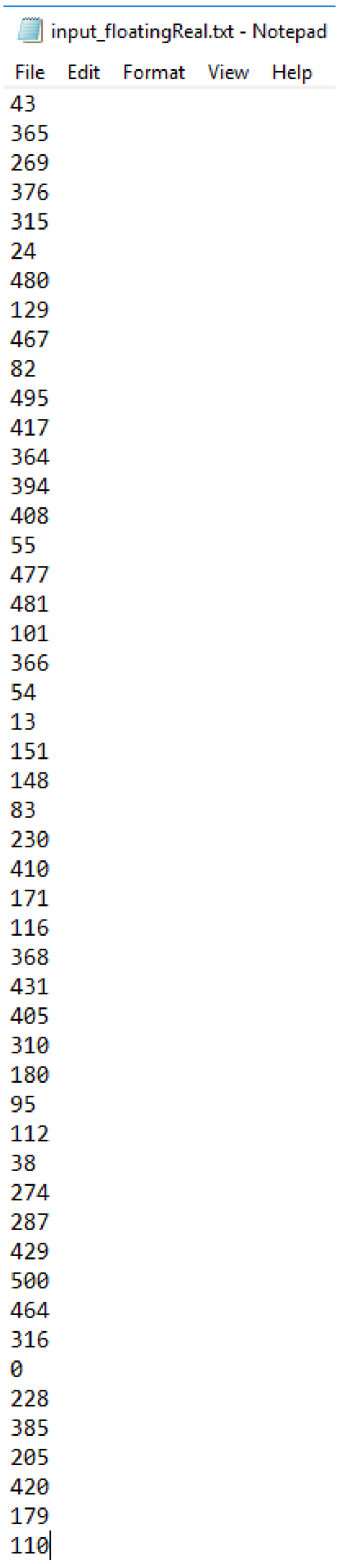
Figure 3: Output For Integer Complex Numbers As Input

float_complexout.txt - Notepad

File Edit Format View Help

```
-394.293978330056 $ -917.236615618077
1486.24092585602 $ -413.427827584198
-1275.19352333377 $ 1492.39974564758
-642.133610919904 $ 36.5285041939520
604.375136295333 $ -566.743704316595
-384.504286488376 $ 1182.72591073215
-1154.59305847736 $ -421.887293951124
323.036400133849 $ 1554.28500178386
-461.606967075186 $ - 1888.40395205162
959.082172493056 $ 1003.93666717407
-882.438427727607 $ -1329.23228103844
-2023.66860739028 $ -733.727391654711
1751.19423074555 $ -566.994816780796
629.251293298174 $ -68.9349197472861
-2259.93704296279 $ -71.9229671118823
-1083.10895941491 $ 1816.56855514635
174.498757084428 $ - 626.368900991037
-2590.62723128634 $ 1322.12921001115
-619.000000000000 $ 128.000000000000
381.198251825687 $ 109.732363768243
302.698360614686 $ 501.783098426547
1057.58922044532 $ 1527.68344300514
-593.518778679489 $ -513.422626990863
1115.37515134291 $ 825.092787157243
-1149.96102909039 $ 578.599800551132
1211.43831036296 $ -1784.32728243514
665.066313612006 $ -177.279294016356
576.643101696344 $ -1096.02120318646
-1704.30506239337 $ -241.804165345742
-816.868163747058 $ -1280.94723514531
1777.87280400399 $ -33.3569191239944
2183.51191266229 $ -159.843310385875
104.190887546521 $ -614.156140490604
-1063.49283372118 $ -192.686371603909
691.141118956884 $ -214.109016987594
879.122230681935 $ 654.003719857377
-51.4818598437252 $ 149.352228419853
91.4326924684502 $ 717.504216835858
1078.14305203795 $ 1021.10684263355
1718.83198891961 $ 754.920743119239
756.736443981558 $ 878.085553073900
1147.79783554787 $ 1121.35052874577
-291.904808705419 $ -1764.31166791979
```

Figure 4: Output For Floating Complex Numbers As Input

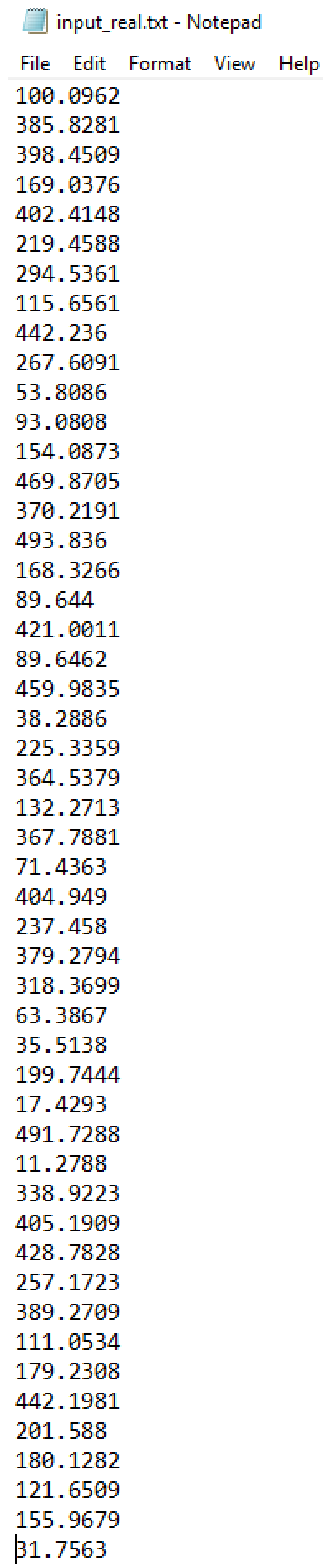


input_floatingReal.txt - Notepad

File Edit Format View Help

43
365
269
376
315
24
480
129
467
82
495
417
364
394
408
55
477
481
101
366
54
13
151
148
83
230
410
171
116
368
431
405
310
180
95
112
38
274
287
429
500
464
316
0
228
385
205
420
179
110

Figure 5: Real Integer Numbers As Input



input_real.txt - Notepad

File Edit Format View Help

100.0962
385.8281
398.4509
169.0376
402.4148
219.4588
294.5361
115.6561
442.236
267.6091
53.8086
93.0808
154.0873
469.8705
370.2191
493.836
168.3266
89.644
421.0011
89.6462
459.9835
38.2886
225.3359
364.5379
132.2713
367.7881
71.4363
404.949
237.458
379.2794
318.3699
63.3867
35.5138
199.7444
17.4293
491.7288
11.2788
338.9223
405.1909
428.7828
257.1723
389.2709
111.0534
179.2308
442.1981
201.588
180.1282
121.6509
155.9679
31.7563

Figure 6: Real Floating Numbers As Input

int_realout.txt - Notepad

File Edit Format View Help

```
13220.0000000000 $ +0.00000000000000
630.635438265233 $ +407.618236021087
-1369.08179755415 $ +335.520466072947
-541.838548014215 $ -1212.28766438599
-28.7810872872226 $ +1727.92076469485
939.507537498111 $ -212.372675800838
-294.786104285329 $ -714.814824326498
-2064.99349074470 $ +633.946241535645
-74.5631890024023 $ +312.052125873080
-365.180553410012 $ +1032.75344093434
-44.3643785156579 $ +1125.75385358348
-355.915442141025 $ -581.306386542235
-20.1414292145151 $ +155.016645418590
-424.551372817546 $ -727.040377454526
-593.103909279808 $ +256.585795134900
640.992462501889 $ -94.8108462845471
95.8288246024455 $ +1815.54057848262
-384.934400255232 $ -354.056498166657
1011.60389074311 $ +24.3962997149673
717.872754405112 $ +1040.95665615670
-743.135621484342 $ -141.472651568959
-590.508298269506 $ -216.356853398543
-1501.90024743448 $ +74.8200009961659
-750.586087018112 $ -1107.96733170895
364.925048712344 $ -1084.03267972697
424.000000000000 $ +0.00000000000000
364.925048712344 $ +1084.03267972697
-750.586087018112 $ +1107.96733170895
-1501.90024743448 $ -74.8200009961659
-590.508298269506 $ +216.356853398543
-743.135621484342 $ +141.472651568959
717.872754405112 $ -1040.95665615670
1011.60389074311 $ -24.3962997149673
-384.934400255232 $ +354.056498166657
95.8288246024455 $ -1815.54057848262
640.992462501889 $ +94.8108462845471
-593.103909279808 $ -256.585795134900
-424.551372817546 $ +727.040377454526
-20.1414292145151 $ -155.016645418590
-355.915442141025 $ +581.306386542235
-44.3643785156579 $ -1125.75385358348
-365.180553410012 $ -1032.75344093434
-74.5631890024023 $ -312.052125873080
-2064.99349074470 $ -633.946241535645
-294.786104285329 $ +714.814824326498
939.507537498111 $ +212.372675800838
-28.7810872872226 $ -1727.92076469485
-541.838548014215 $ +1212.28766438599
-1369.08179755415 $ -335.520466072947
630.635438265233 $ -407.618236021087
```

Figure 7: Output For Integer Real Numbers As Input

float_realout.txt - Notepad

File Edit Format View Help

```
1.2261 $ 0.0000
-0.0102 $ -0.0241
-0.0541 $ 0.0242
-0.0609 $ -0.0655
0.0481 $ -0.1714
-0.0482 $ 0.0163
-0.0904 $ -0.0017
0.0848 $ -0.0655
-0.0102 $ -0.1262
-0.0463 $ -0.0956
0.0202 $ 0.1440
0.0115 $ -0.0125
-0.0193 $ -0.0587
-0.0009 $ -0.0379
-0.0831 $ -0.0757
-0.0716 $ -0.0642
-0.0173 $ -0.0533
0.0934 $ 0.0287
-0.0433 $ 0.0415
-0.1096 $ 0.0035
0.1077 $ -0.0489
-0.2281 $ -0.0067
0.0218 $ 0.0691
0.0779 $ -0.0684
0.0919 $ 0.1308
-0.0529 $ 0.0000
0.0919 $ -0.1308
0.0779 $ 0.0684
0.0218 $ -0.0691
-0.2281 $ 0.0067
0.1077 $ 0.0489
-0.1096 $ -0.0035
-0.0433 $ -0.0415
0.0934 $ -0.0287
-0.0173 $ 0.0533
-0.0716 $ 0.0642
-0.0831 $ 0.0757
-0.0009 $ 0.0379
-0.0193 $ 0.0587
0.0115 $ 0.0125
0.0202 $ -0.1440
-0.0463 $ 0.0956
-0.0102 $ 0.1262
0.0848 $ 0.0655
-0.0904 $ 0.0017
-0.0482 $ -0.0163
0.0481 $ 0.1714
-0.0609 $ 0.0655
```

Figure 8: Output For Floating Real Numbers As Input