ICT1054

Data Structures and Algorithms

University of Malta

B.Sc. IT (Hons.) Computing and Business

Luigi Naudi

0249603(L)

Contents:

Documentation:

Source Code Listing:

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as “the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I / We*, the undersigned, declare that the [assignment / Assigned Practical Task report / Final Year Project report] submitted is my / our* work, except where acknowledged and referenced.

I / We* understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Luigi Naudi	
Student Name	Signature
Student Name	Signature
Student Name	Signature
Student Name	Signature

ICT 1018	Data Structures and Algorithms Assignment
Course Code	Title of work submitted

05/1/2022

Date

Statement of Completion:

Question 1 - Attempted and Works Well

Question 2 – Attempted and Works Well

Question 3 – Attempted and Works Well

Question 4 – Attempted and Works Well

Question 5 – Completed but has some bugs:

- Crashes when coming across invalid char or a non RPN expression

Question 6 – Attempted and Works Well

Question 7 – Attempted and Works Well

- Ran into difficulty displaying BST

Question 8 – Attempted and Works Well

Question 9 – Attempted and Works Well

Question 10 – Attempted and Works Well

Question 11 – Attempted and Works Well

Question 12 – Attempted and Works Well

Question 1

References: [1] [2]

The program creates two arrays of random sizes 256 and 300 containing randomly generated numbers between 0 and 1024.Array A is sorted using Shell sort, whilst array B is sorted using Quick sort. The output for Array A shows the unsorted array, and the sorted array after Shell sort has been implemented. The output for array B also shows the unsorted array, and the sorted array after Quick sort has been implemented.

```
PS C:\Users\luigi\Desktop\Assignments Semester
signments Semester 2\ICT1018 - Assignment\Q1and

Array A Before Shell Sort:
  [101, 573, 994, 681, 450, 362, 139, 410, 524,
 881, 660, 222, 617, 992, 355, 453, 899, 212, 4
57, 849, 602, 494, 3, 851, 972, 734, 564, 914,
09, 728, 171, 792, 705, 572, 182, 443, 727, 589
, 1003, 609, 158, 529, 649, 83, 360, 437, 400,
74, 11, 776, 368, 29, 1006, 27, 356, 801, 650,
8, 203, 416, 248, 681, 799, 147, 462, 433, 838,
172, 964, 684, 61, 40, 711, 495, 725, 744, 609,
571, 520, 414, 768, 47, 63, 654, 6, 419, 202, 5
8, 678, 409, 778, 337, 38, 271, 70, 593, 958, 4
15, 612, 321, 854, 147, 255, 599, 229, 180, 309
, 176, 123, 268, 305, 475, 881, 496, 481, 883,

Array A After Shell Sort:
  [3, 4, 6, 11, 16, 26, 27, 29, 38, 40, 42, 47,
, 123, 126, 129, 138, 139, 139, 141, 145, 147,
182, 183, 202, 203, 208, 209, 212, 222, 228, 22
8, 271, 274, 274, 277, 284, 287, 293, 294, 298,
 360, 360, 362, 366, 368, 383, 388, 394, 395, 4
40, 443, 450, 452, 453, 460, 462, 475, 475, 481
, 529, 538, 539, 539, 560, 564, 571, 572, 573,
617, 617, 621, 621, 630, 641, 642, 649, 650, 65
3, 693, 697, 698, 700, 705, 711, 717, 722, 725,
 761, 765, 768, 775, 776, 778, 783, 788, 788, 7
51, 854, 857, 863, 867, 871, 875, 877, 881, 881
, 972, 973, 975, 980, 986, 991, 992, 994, 995,
```

```
Array B Before Quick Sort:
  [711, 169, 859, 420, 571, 967, 80, 32, 240, 4
9, 379, 511, 537, 51, 451, 549, 100, 428, 319,
547, 532, 8, 643, 591, 774, 221, 821, 821, 516,
377, 368, 654, 180, 106, 378, 330, 245, 74, 367
, 487, 425, 525, 851, 420, 377, 441, 473, 277,
175, 484, 621, 341, 706, 576, 211, 461, 282, 35
 648, 456, 567, 352, 281, 363, 615, 96, 107, 47
, 341, 654, 560, 637, 971, 324, 819, 246, 714,
78, 473, 633, 441, 551, 569, 816, 645, 263, 798
, 723, 541, 334, 157, 10, 41, 555, 128, 469, 31
, 325, 640, 456, 387, 547, 227, 11, 495, 163, 6
83, 139, 109, 315, 309, 301, 605, 868, 575, 662

Array B After Quick Sort:
  [1, 2, 4, 8, 10, 11, 16, 20, 22, 25, 32, 34,
180, 180, 183, 197, 211, 211, 211, 219, 219, 22
309, 309, 313, 313, 314, 315, 316, 319, 320, 32
429, 431, 438, 441, 441, 441, 451, 456, 456, 46
531, 532, 537, 538, 539, 541, 546, 547, 547, 54
634, 637, 638, 640, 642, 643, 645, 646, 648, 65
760, 774, 774, 776, 776, 781, 781, 798, 807, 80
948, 953, 962, 967, 967, 971, 977, 978, 979, 98
```

Question 2

Since Question 2 is a continuation of Question 1, it was made sure that it works well before moving on. An empty array C was created as well as two pointers, one for array A and one for array B. The pointers are used in the algorithm to decide which element from both arrays is smallest. The smallest element is added to array C whilst the pointer of the smaller element is incremented, and compared to the other pointer once again, and so on. The pointers allow for the merged array C to be sorted from the beginning of the populating process. Looking at the previous results of array A and B it is evident that the merging was is successful and sorted as expected.

```
Array C:
 [1, 2, 3, 4, 4, 6, 8, 10, 11, 11, 16, 16, 20,
109, 112, 115, 116, 119, 120, 121, 122, 123, 12
176, 178, 180, 180, 180, 182, 183, 183, 197, 20
248, 250, 252, 253, 255, 259, 261, 262, 263, 26
313, 314, 315, 316, 319, 320, 321, 322, 324, 32
378, 378, 379, 380, 383, 383, 387, 387, 388, 38
453, 456, 456, 460, 461, 462, 462, 462, 464, 46
511, 514, 514, 516, 520, 521, 524, 525, 526, 52
573, 574, 574, 575, 576, 580, 581, 587, 589, 59
638, 640, 641, 642, 642, 643, 645, 646, 648, 64
697, 698, 699, 700, 701, 705, 706, 708, 711, 71
768, 774, 774, 775, 776, 776, 776, 778, 781, 78
851, 854, 857, 859, 863, 864, 865, 865, 867, 86
958, 958, 962, 964, 967, 967, 971, 972, 973, 97
1022]
```

Question 3

There are no extreme points in a sorted array since element (x) will always be larger than element (x-1) and less than element (x+1), thus, to test this question I had to use both an unsorted array and a sorted array. It was chosen not to randomly generate the numbers of the array since the likelihood of it being sorted is way less than that of it being sorted.

When using a sorted array, the output shows that it has been recognised that there are no extreme points in the array:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q3.py"
Array:
 [2, 5, 7, 9, 23, 53, 67]
No Extreme Elements are in the Array
```

On the other hand, when using an unsorted array, the output indicated clearly that there are extreme points in the array and can identify them clearly.

```
No Extreme Elements are in the Array
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q3.py"
Array:
 [0, 5, 3, 6, 8, 7, 15, 9]

Extreme Elements:

[5, 3, 8, 7, 15]
```

This indicates that the algorithm can cater for both types of arrays and has an expected and therefore correct output.

Question 4

For this Question an array of 30 random numbers is generated. Then, the product of each possible pair without repetition is calculated and placed into another array in the format [element A, element B, their product]. Two variables store and comp hold the products of two elements in the array and are compared. If they are equal, they are pushed to another array.

The problem was duplicate elements in the pairs array, which lead to duplicate outputs in the end. To solve this, the array is pushed into a hash set (then converted back into an array) which would filter and remove duplicate elements from the array.

The output size differs from size due to the array being randomly generated. All scenarios were tested such as a no two-pair array, and an array with many duplicates:

No two-pair array output:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desk
top\Assignments Semester 2\ICT1018 - Assignment\Q4.py"

Array :
 [2, 5]

There are no Two-Pairs in this Array
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

Duplicate filled array:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semes
ter 2\ICT1018 - Assignment\Q4.py"

Array :
 [2, 5, 62, 46, 1, 42, 75, 24, 8, 35, 13, 11, 23, 9, 2, 5, 62, 46, 1, 42, 75, 24, 8, 35, 13, 11, 23, 9]

1 & 46 and 2 & 23 are a Two-Pair
```

Random Generated Array:

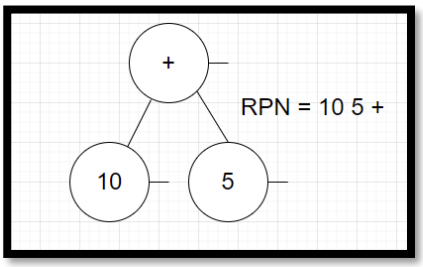
```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q4.py"

Array :
 [116, 41, 70, 104, 8, 59, 19, 67, 38, 99, 83, 30, 76, 15, 46, 6, 122, 22, 122, 118, 87, 43, 44, 57, 108, 61, 96, 30, 27, 51]

6 & 76 and 8 & 57 are a Two-Pair

6 & 99 and 22 & 27 are a Two-Pair

6 & 116 and 8 & 87 are a Two-Pair

15 & 38 and 19 & 30 are a Two-Pair

15 & 44 and 22 & 30 are a Two-Pair

15 & 76 and 30 & 38 are a Two-Pair

15 & 118 and 30 & 59 are a Two-Pair

15 & 122 and 30 & 61 are a Two-Pair

19 & 44 and 22 & 38 are a Two-Pair

19 & 108 and 27 & 76 are a Two-Pair

19 & 118 and 38 & 59 are a Two-Pair

19 & 122 and 38 & 61 are a Two-Pair

22 & 76 and 38 & 44 are a Two-Pair

22 & 118 and 44 & 59 are a Two-Pair

22 & 122 and 44 & 61 are a Two-Pair

38 & 118 and 59 & 76 are a Two-Pair

38 & 122 and 61 & 76 are a Two-Pair

57 & 116 and 76 & 87 are a Two-Pair

59 & 122 and 61 & 118 are a Two-Pair
```

Apart from these, tests were made with smaller two-pair arrays to check if the correctness of output.

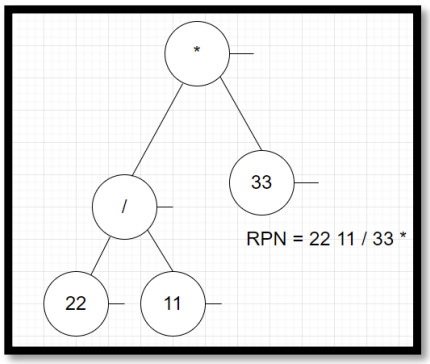
```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q4.py"

Array :
 [2, 5, 10, 13, 2, 12, 8, 4, 6]

2 & 10 and 4 & 5 are a Two-Pair

2 & 12 and 4 & 6 are a Two-Pair

4 & 10 and 5 & 8 are a Two-Pair

4 & 12 and 6 & 8 are a Two-Pair

5 & 12 and 6 & 10 are a Two-Pair
```

Question 5

This Question was tested with different RPN's of varying sizes. Result was as expected:




```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> pytho
n -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q5.
py"
[]
[10]
[10, 5]
15
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

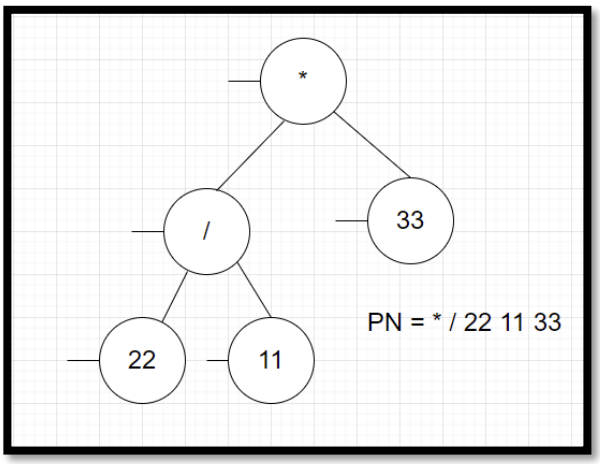



```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> pytho
n -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q5.
py"
[]
[22]
[22, 11]
[2.0]
[2.0, 33]
66.0
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

It was also tested using an incorrect input 'b' which leads the program to crash since it is not programmed to handle anything other than integers and simple arithmetic operators:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> pytho
n -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q5.
py"
[]
[100]
[100, 2]
[102]
[102, 30]
Traceback (most recent call last):
  File "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q
5.py", line 36, in <module>
    print(Filler("100 2 + 30 b"))
  File "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q
5.py", line 27, in Filler
    stack.append(int(op))
ValueError: invalid literal for int() with base 10: 'b'
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

Finally, it was tested with a previous mathematic sum used, but in PN format instead:




```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> pytho
n -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q5.
py"
[]
Traceback (most recent call last):
  File "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q
5.py", line 37, in <module>
    print(Filler("* / 22 11 33"))
  File "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q
5.py", line 12, in Filler
    num2 = stack.pop()
IndexError: pop from empty list
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

As expected, the program crashed here too. This is because the calculator is only programmed to handle RPN expressions

Question 6

References: [3]

The first Question was tested with a prime number (3) and a non-prime number (12). After verifying that it works, A random number was used to test and clarify that the algorithm produces the expected output.

Prime Number Test:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desk
top\Assignments Semester 2\ICT1018 - Assignment\Q6.py"
3 is Prime
```

Non-Prime Number Test:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desk
top\Assignments Semester 2\ICT1018 - Assignment\Q6.py"
12 is NOT Prime and can be divided: 2
```

Random Number Test:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desk
top\Assignments Semester 2\ICT1018 - Assignment\Q6.py"
201 is NOT Prime and can be divided: 3
```

For the Second Question testing, inputs from limits between 10 – 60 were randomly picked. Naturally if the input 60 were to produce the correct output, all limits under that number should produce the expected result up to that limit also. Nonetheless all the limits were tested for certainty and the outputs were as expected.

```
Following are the prime numbers smaller than or equal to 60
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

Question 7

References: [4]

For this question the requirements were clearly listed to plan. Displaying the BST visually was a problem I ran into, thus testing if a proper tree could be built was difficult.
Initially it was checked to see if the algorithm could properly handle a root node and a left and right node for it:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -
u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q7.py"
Enter Root Node:
15
Enter A New Node Into The BST Or Press Enter To Exit: 6
Enter A New Node Into The BST Or Press Enter To Exit: 18
Enter A New Node Into The BST Or Press Enter To Exit:
Exiting Input Procedure...
Root:  15
Left  6
Right 18
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

After this, an algorithm to visually represent the BST was created. It had some flaws at first due to recursion and the .format method disagreeing with each other and producing an expected but unwanted output.

With three simple input it works well:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q7.py"
Enter Root Node:
100
Enter A New Node Into The BST Or Press Enter To Exit: 50
Enter A New Node Into The BST Or Press Enter To Exit: 150
Enter A New Node Into The BST Or Press Enter To Exit:
Exiting Input Procedure...
                                    100
                                50            150
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

As inputs grew, the algorithm showed its flaws and was not displaying the BST the way it was intended:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q7.py"
Enter Root Node:
100
Enter A New Node Into The BST Or Press Enter To Exit: 50
Enter A New Node Into The BST Or Press Enter To Exit: 150
Enter A New Node Into The BST Or Press Enter To Exit: 15
Enter A New Node Into The BST Or Press Enter To Exit: 12
Enter A New Node Into The BST Or Press Enter To Exit: 160
Enter A New Node Into The BST Or Press Enter To Exit: 101
Enter A New Node Into The BST Or Press Enter To Exit:
Exiting Input Procedure...
                                    100
                                50            150
                                15
                                12
                               101            160
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

The problem was that the .format spacing was not being changed after each method call, thus the numbers after the second level of the BST being printed out underneath each other. Apart from that, as seen above, 101 is placed under the left side of the subtree, this is because Node 101 is still 150's leftNode, but the display algorithm has no way of realizing that to place it under 150.

Although the above can easily be implemented, still, the program would have no way to indent for each level and cater for each subtree. The algorithm was scrapped but included in the code as a comment.

The code from line 48-84 was inspired by [4] and works well. It was tested using a range of outputs and produced results as expected:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q7.py"
Enter Root Node:
100
Enter A New Node Into The BST Or Press Enter To Exit: 50
Enter A New Node Into The BST Or Press Enter To Exit: 150
Enter A New Node Into The BST Or Press Enter To Exit: 45
Enter A New Node Into The BST Or Press Enter To Exit: 56
Enter A New Node Into The BST Or Press Enter To Exit: 145
Enter A New Node Into The BST Or Press Enter To Exit: 180
Enter A New Node Into The BST Or Press Enter To Exit: 200
Enter A New Node Into The BST Or Press Enter To Exit: 96
Enter A New Node Into The BST Or Press Enter To Exit:
Exiting Input Procedure...
              100
       50               150
   45       56       145       180
             96               200
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q7.py"
Enter Root Node:
85
Enter A New Node Into The BST Or Press Enter To Exit: 455
Enter A New Node Into The BST Or Press Enter To Exit: 96
Enter A New Node Into The BST Or Press Enter To Exit: 85
Enter A New Node Into The BST Or Press Enter To Exit: 47
Enter A New Node Into The BST Or Press Enter To Exit: 89
Enter A New Node Into The BST Or Press Enter To Exit: 300
Enter A New Node Into The BST Or Press Enter To Exit: 960
Enter A New Node Into The BST Or Press Enter To Exit: 1000
Enter A New Node Into The BST Or Press Enter To Exit: 975
Enter A New Node Into The BST Or Press Enter To Exit:
Exiting Input Procedure...
                          85
              47                          455
                                96                960
   85       300              1000
     89           975
```

Question 8

References: [5]

This Question's accuracy is dependent on the number of iterations the algorithm goes through. To test, different numbers were inputted to calculate an approximation of their square root.

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q8.py"
With  10  iterations:

Approximate Square Root for 9:  3.0

Approximate Square Root for 2:  1.4142135623730951

Approximate Square Root for 12:  3.4641016151377544

Approximate Square Root for 37:  6.082762530298219

Approximate Square Root for 53:  7.280109889280518

Approximate Square Root for 9:  5.0
```

The result was compared to an online calculator to check the accuracy and correctness of the output. The results were as expected.

For this question it was particularly interesting testing different iterations and noticing the change in accuracy of calculations:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q8.py"
With  5  iterations:

Approximate Square Root for 9:  3.000000001396984

Approximate Square Root for 2:  1.414213562373095

Approximate Square Root for 12:  3.4641016533502986

Approximate Square Root for 37:  6.08306027903096

Approximate Square Root for 53:  7.282204131703878

Approximate Square Root for 9:  5.000023178253949

PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q8.py"
With  3  iterations:

Approximate Square Root for 9:  3.023529411764706

Approximate Square Root for 2:  1.4142156862745097

Approximate Square Root for 12:  3.5243264799716414

Approximate Square Root for 37:  7.003173763554615

Approximate Square Root for 53:  9.07066401439803

Approximate Square Root for 9:  5.406026962727994

PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q8.py"
With  1  iterations:

Approximate Square Root for 9:  5.0

Approximate Square Root for 2:  1.5

Approximate Square Root for 12:  6.5

Approximate Square Root for 37:  19.0

Approximate Square Root for 53:  27.0

Approximate Square Root for 9:  13.0
```

Question 9

For this question a similar approach to Q4 was taken using sets to locate repeated elements. It was tested using two different arrays, one with many repeated elements and another with no repeated elements.

Testing using an array with repeated elements. Expected output was:
[1, 2, 3, 6, 7, 8, 9, 11, 13, 51]
Result was correct and as expected

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q9.py"
Array:
 [1, 2, 3, 13, 5, 6, 51, 7, 30, 8, 12, 9, 10, 11, 1, 2, 3, 4, 6, 13, 7, 8, 9, 11, 50, 51]

The repeated values in the Array are:
 [1, 2, 3, 6, 7, 8, 9, 11, 13, 51]
```

Testing using an array with no repeated elements. The expected output was correct:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q9.py"
Array:
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 24, 67, 41, 72]

There are no repeated elements in this Array
```

Question 10

References: [6]

To test this a list of random numbers was used, with the largest number known, the expected output was 55, showing that the result below was correct:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q10.py"
[1, 2, 55, 3, 13, 5, 6, 30, 8, 14] :

Largest Number in List:  55
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

It was also tested using an array of the same number only, making the expected output to be 5, and as seen below, correct:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q10.py"
[5, 5, 5, 5, 5, 5] :

Largest Number in List:  5
```

The last test checked if the program could handle negative numbers and 0, which as expected produced the correct result once again:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q10.py"
[0, -2, -3, -1, -55, -9] :

Largest Number in List:  0
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

Question 11

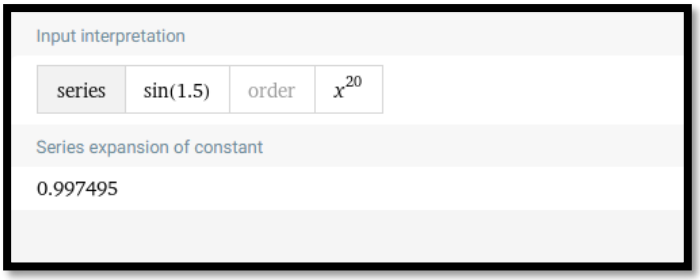
References [7] [8], were used to obtain and implement the formulas for sin and cos.

This Question was tested using different values for n (terms of the expansion) as well as different values of x for cos(x) and sin(x). The outputs were compared to an online calculator: https://www.wolframalpha.com/calculators/series-calculator

Testing using n = 20 and setting x to 1.5

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop
\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the order (n) for the expansion:
20
Enter the value of x:
1.5
Cos( 1.5 ) to order 20 = 0.0707372016677029

Sin( 1.5 ) to order 20 = 0.9974949866040546
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

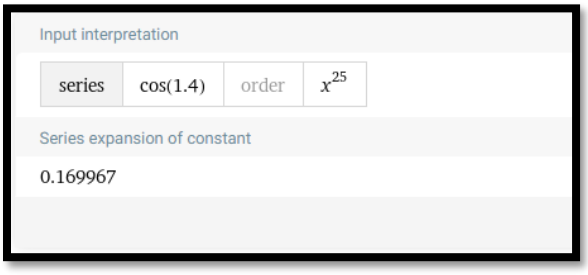
Confirming result is therefore, as expected:

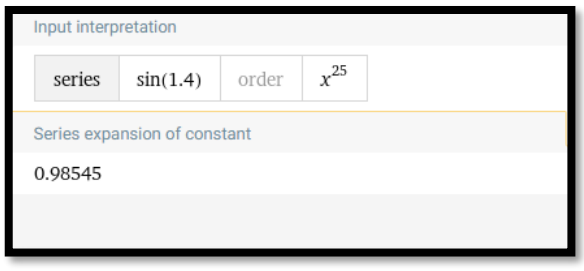


Testing using n = 25 and setting x to 1.4

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\Desktop
\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the order (n) for the expansion:
25
Enter the value of x:
1.4
Cos( 1.4 ) to order 25 = 0.16996714290024104

Sin( 1.4 ) to order 25 = 0.9854497299884603
```

Once again result was as expected:





Question 12

For this Question the code to generate the Fibonacci sequence was implemented. It was tested using a random number and the output was as expected.

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the value for n:
12
The Fibonacci Sequence up to the first 12 elements is :
 [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

When input n was set to 0 or 1, the output was not as expected:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\xampp\htdocs\
Assignment\CIS1054-Part1\tempCodeRunnerFile.python"
Enter the value for n:
0
[0, 1]
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\xampp\htdocs\
Assignment\CIS1054-Part1\tempCodeRunnerFile.python"
Enter the value for n:
1
[0, 1]
```

This was because the program had to way of detecting 0 and 1 and always printed the initial array [0,1]. The program was changed to cater for such inputs and output was as expected:

```
n -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q11
.py"
Enter the value for n:
0
!! n cannot be equal to or less than 0 !!
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

```
n -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q11
.py"
Enter the value for n:
1
The Fibonacci Sequence up to the first 1 elements is :
 [0]
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> pytho
n -u "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q11
.py"
Enter the value for n:
2
The Fibonacci Sequence up to the first 2 elements is :
 [0, 1]
```

After the first part of the program was confirmed to function as expected the sum of each element of the Fibonacci sequence was implemented and tested using, once again, 0,1,2, and 12:

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the value for n:
0
!! n cannot be equal to or less than 0 !!
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the value for n:
1
The Fibonacci Sequence up to the first 1 elements is :
 [0]

The Fibonacci Sequence SUM of the first 1 elements is: 0
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the value for n:
2
The Fibonacci Sequence up to the first 2 elements is :
 [0, 1]

The Fibonacci Sequence SUM of the first 2 elements is: 1
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the value for n:
12
The Fibonacci Sequence up to the first 12 elements is :
 [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

The Fibonacci Sequence SUM of the first 12 elements is: 232
```

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the value for n:
10
The Fibonacci Sequence up to the first 10 elements is :
 [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

The Fibonacci Sequence SUM of the first 10 elements is: 88
```

It was also tested with an invalid input such as a letter. Here the program crashes as it is not programmed to take anything other than an integer value

```
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment> python -u "c:\Users\luigi\D
esktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py"
Enter the value for n:
x
Traceback (most recent call last):
  File "c:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment\Q11.py", line 5, in <mo
dule>
    n = int(input())
ValueError: invalid literal for int() with base 10: 'x'
PS C:\Users\luigi\Desktop\Assignments Semester 2\ICT1018 - Assignment>
```

References:

[1]: “ShellSort.” *GeeksforGeeks*, 16 June 2014, www.geeksforgeeks.org/shellsort/

[2]: “Python Program for QuickSort.” *GeeksforGeeks*, 7 Jan. 2014, www.geeksforgeeks.org/python-program-for-quicksort/

[3]: “Python Program for Sieve of Eratosthenes.” *GeeksforGeeks*, 27 July 2012, www.geeksforgeeks.org/python-program-for-sieve-of-eratosthenes/

[4]: “Print Binary Tree Level by Level in Python.” *Stack Overflow*, stackoverflow.com/questions/34012886/print-binary-tree-level-by-level-in-python.

[5]: “Find Root of a Number Using Newton’s Method.” *GeeksforGeeks*, 7 Feb. 2020, www.geeksforgeeks.org/find-root-of-a-number-using-newtons-method/#:~:text=Let%20N%20be%20any%20number. Accessed 1 May 2022.

[6]: “Recursion - Python: Recursive Function to Find the Largest Number in the List.” *Stack Overflow*, stackoverflow.com/questions/12711397/python-recursive-function-to-find-the-largest-number-in-the-list. Accessed 1 May 2022.

[7]: “Math - Series Expansion of Cos with Python.” *Stack Overflow*, stackoverflow.com/questions/42302023/series-expansion-of-cos-with-python. Accessed 1 May 2022.

[8]: “How to Calculate Taylor Series and Lewis Carrol Divisbilty Test in Python 3.5 without Using the Math Module.” *Stack Overflow*, stackoverflow.com/questions/38002712/how-to-calculate-taylor-series-and-lewis-carrol-divisbilty-test-in-python-3-5-wi. Accessed 1 May 2022.

Source Code Listing

Question 1 and 2:

```
import random as rd
from turtle import position

#Two Arrays of Unequal size (256 and 300) containing randomly generated numbers
between 0 and 1024
A = [rd.randrange(0,1025)for i in range(256)]
B = [rd.randrange(0,1025)for i in range(300)]
C = []

#Shell Sort Algorithm Method
def shell_Sort(array):
    length = len(array)
    gap = length//2

    while gap > 0:
        for i in range(gap, length):
            temp = array[i]
            j = i
            while j >= gap and array[j - gap] > temp:
                array[j] = array[j - gap]
                j -= gap

            array[j] = temp
        gap //= 2

#Quick Sort Algorithm Method
def split(array, low, high):
    idx_small = (low-1)
    pivot = array[high]

    for i in range(low, high):
        # Current element = array[j]
        if array[i] <= pivot:

            # Increment index of smaller element
            idx_small = idx_small+1
            array[idx_small], array[i] = array[i], array[idx_small]

    array[idx_small+1], array[high] = array[high], array[idx_small+1]
```

```
    return (idx_small+1)

def quick_Sort(array, low, high):
    if len(array) == 1:
        return array
    if low < high:

        # Index for Split
        spin = split(array, low, high)

        # Sorting elements before and after splitting them up
        quick_Sort(array, low, spin-1)
        quick_Sort(array, spin+1, high)

#Printing Array A before and After using Shell Sort
print("\nArray A Before Shell Sort: \n " ,A)
shell_Sort(A)
print("\nArray A After Shell Sort: \n " ,A)


#Printing Array B before and After using Quick Sort
print("\nArray B Before Quick Sort: \n " ,B)
h = len(B) - 1
quick_Sort(B, 0 ,h)
print("\nArray B After Quick Sort: \n " ,B)

#Method to merge arrays using pointers and loops
def merge_Arrays(A, B, C):
    #Setting the length of Array C
    length = len(A) + len(B)

    pointer_A = 0
    pointer_B = 0
    d = 0

    while (length - 2 > d):
        d = pointer_A + pointer_B
        if (pointer_A == len(A)-1):
            C.append(B[pointer_B])
            pointer_B += 1

        elif (pointer_B == len(B)-1):
            C.append(A[pointer_A])
            pointer_A += 1
```

```
        elif (A[pointer_A] <= B[pointer_B]):
            C.append(A[pointer_A])
            pointer_A += 1

        else:
            C.append(B[pointer_B])
            pointer_B += 1


merge_Arrays(A, B, C)
print("\nArray C: \n" ,C)
```

Question 3:

```
#Extreme Points = Element is bigger/smaller than both previous and following
elements
A = [0, 5, 3, 6, 8, 7, 15, 9]
#A = [2,5,7,9,23,53,67]

#Array to Store all Random Points
extreme_Points = []
size = len(A)

#Printing the Array
print("Array: \n" ,A)

for i in range (0,size):
    if (0 < i < size-1) and ((A[i-1] < A[i] > A[i+1]) or (A[i-1] > A[i] <
A[i+1])):
        extreme_Points.append(A[i])

if(extreme_Points == []):
    print("No Extreme Elements are in the Array")
else:
    print("\nExtreme Elements:\n")
    #Printing the Extreme Elements in the Array
    print(extreme_Points)
```

Question 4:

```
import random as rd

#Creating a Random Array of 30 Numbers
Temp_Set = set()
nums = [rd.randrange(1,124)for i in range(30)]
#nums = [2,5]
#nums = [2,5,10,13,2,12,8,4,6]
#nums =
[2,5,62,46,1,42,75,24,8,35,13,11,23,9,2,5,62,46,1,42,75,24,8,35,13,11,23,9]

print("\nArray : \n" ,nums ,"\n")

for a in nums:
    Temp_Set.add(a)
Array = list(Temp_Set)

size = len(Array)
products = []
pairs = []

for a in range(size):
    #Populating the Array with [(Number A)(Number B)(Their Product)]
    for b in range(a + 1, size):
        product = Array[a]*Array[b]
        products.append([Array[a],Array[b],product])

length = len(products)

#Checking Each Element's Product in the Array and adding Number A and B to Array:
Pairs if Equal
for x in range(0,length -1):
    #store holds product of each element [(0:element A)(1:element
B)(2:product)]
    store = products[x][2]
    for y in range((x+1), length):
        comp = products[y][2]
        #print("A = " +str(products[x]) +" B: " +str(products[y]))
        if(store == comp):
            #print(str(store) +" is Equal to " +str(comp))
            pairA = [products[x][0]],[products[x][1]]
            pairB = [products[y][0]],[products[y][1]]
            pairs.append([[pairA],[pairB]])
```

```
#Printing All The 2-Pairs
if pairs:
    for z in range(0,(len(pairs))):
        print(str(pairs[z][0][0][0][0]) +" & " +str(pairs[z][0][0][1][0]) +" and
" +str(pairs[z][1][0][0][0]) +" & " +str(pairs[z][1][0][1][0]) +" are a Two-
Pair\n")
else: print("There are no Two-Pairs in this Array")
```

Question 5:

```
stack = []

def Filler(data):
    operator  = data.split()

    for op in operator:

        print (stack)

        if op in {"+", "-", "*", "/"}:

            num2 = stack.pop()
            num1 = stack.pop()

            if op == "+":
                ans = num1 + num2
            if op == "-":
                ans = num1 - num2
            if op == "*":
                ans = num1 * num2
            if op == "/":
                ans = num1 / num2

            stack.append(ans)

        else:
            stack.append(int(op))


    return stack.pop()


#Test Data:
```

```
print(Filler("10 5 +"))
print(Filler("22 11 / 33 *"))
#print(Filler("100 2 + 30 b"))
#print(Filler("3 * / 22 11 33"))
```

Question 6:

```
import random as rd
from unicodedata import numeric

def Prime_Checker(num):
    prime = True
    #Only Divisible by 1 and itself
    for i in range(2,num):
        if ((num % i) == 0):
            #not a prime number
            prime = False
            break

    if (prime == True):
        print(str(num) +" is Prime")
    else:
        print(str(num) +" is NOT Prime and can be divided:" ,i)

num = rd.randint(1,1000)
#num = 12
#num = 3
Prime_Checker(num)

def SieveOfEratosthenes(n):
    prime = [True for i in range(n+1)]
    p = 2
    while (p * p <= n):
        # If prime[p] is not changed, then it is a prime
        if (prime[p] == True):

            # Update all multiples of p
            for i in range(p * p, n+1, p):
                prime[i] = False
        p += 1

    # Print all prime numbers
    for p in range(2, n+1):
        if prime[p]:
```

```
            print(p)

# Driver Code to Test different instances of n
nums = [10,20,30,40,50,60]
i = rd.randint(0,(len(nums)-1))
n = nums[i]
print("Following are the prime numbers smaller than or equal to", n)
SieveOfEratosthenes(n)
```

Question 7:

```
'''
#Create a non balanced BST with inputs one at a time
#Needs Class, Object Node with left and right, methods for adding node to tree
and possible visualisation using JSON

DIFFICULTY TO DISPLAY BST
'''
class Node:
    def __init__(self, num):
        self.leftNode = None
        self.rightNode = None
        self.num = num

def addNode(root, node):
    if (root == None):
        node = root
    else:
        if (node.num < root.num):
            #place on left if not null
            if(root.leftNode == None):
                root.leftNode = node
            #Recursion to Add on Left of Left Nodes below Root
            else:
                addNode(root.leftNode, node)
        #else if(node.num > root.num)
        else:
            if(root.rightNode == None):
                root.rightNode = node
            #Recursion to Add on Left of Left Nodes below Root
            else:
                addNode(root.rightNode, node)

stop_input = False
```

```
print("Enter Root Node: ")
root = Node(int(input()))

#While loop to Ask User to input Node value
while stop_input == False:
    value = input("Enter A New Node Into The BST Or Press Enter To Exit: ")

    #Ignore Case for Exit value
    if not value:
        print("Exiting Input Procedure...")
        stop_input = True
    else:
        num = int(value)
        addNode(root, Node(num))

def bfs(node,level=0,res=[]):
  if level<len(res):
    if node:
      res[level].append(node.num)
    else:
      res[level].append(" ")
  else:
    if node:
      res.append([node.num])
    else:
      res.append([" "])
  if not node:
    return
  bfs(node.leftNode,level+1,res)
  bfs(node.rightNode,level+1,res)
  return res

def printBST(node):
  treeArray = bfs(node)
  h = len(treeArray)
  whiteSpaces = (2**h)-1

  def printSpaces(n):
    for i in range(n):
      print(" ",end="")


  for level in treeArray:
    whiteSpaces = whiteSpaces//2
    for i,x in enumerate(level):
```

```
      if i==0:
        printSpaces(whiteSpaces)
      print(x,end="")
      printSpaces(1+2*whiteSpaces)
    print()
#driver Code
printBST(root)

'''
For Testing Purposes Part 1
print("Root: " ,root.num)
print("Left " ,root.leftNode.num)
print("Right" ,root.rightNode.num)
'''

'''
Part2
def display_node(node):
    if (node.leftNode and node.rightNode):
        #for every level of depth -5
        print("{:>40}".format(node.leftNode.num),"{:>20}".format(node.rightNode.n
um))
        display_node(node.leftNode)
        display_node(node.rightNode)

    elif(node.leftNode):
        print("{:>40}".format(node.leftNode.num))
        display_node(node.leftNode)

    elif(node.rightNode):
        print("{:>60}".format(node.rightNode.num))
        display_node(node.rightNode)


print ("{:>48}".format(root.num))
display_node(root)
'''
```

Question 8:

```
def newtonRaphson(n, iter):

    a = float(n) # number to get square root of
    for i in range(iter):
        n =(n + a / n)/2
    return n

#number of iterataions (effects the accuracy of the final answer)
iter = 1

"""
Testing Task 8
"""
print("With " ,iter ," iterations: \n")
print ("Approximate Square Root for 9: " ,newtonRaphson(9,iter) ,"\n")
print ("Approximate Square Root for 2: " ,newtonRaphson(2,iter) ,"\n")
print ("Approximate Square Root for 12: " ,newtonRaphson(12,iter) ,"\n")
print ("Approximate Square Root for 37: " ,newtonRaphson(37,iter) ,"\n")
print ("Approximate Square Root for 53: " ,newtonRaphson(53,iter) ,"\n")
print ("Approximate Square Root for 9: " ,newtonRaphson(25,iter) ,"\n")
```

Question 9:

```
array = [1,2,3,13,5,6,51,7,30,8,12,9,10,11,1,2,3,4,6,13,7,8,9,11,50,51]
#repeated should be 1, 2, 3, 6, 7, 8, 9, 11, 13, 51
'''
array = [1,2,3,4,5,6,7,8,9,10,11,24,67,41,72]
should have no repeated elements
'''
print("Array: \n" ,array)
repeated = set()
not_repeated = set()

for x in array:
    current_length = len(not_repeated)
    not_repeated.add(x)
    #After Adding Each element to set, I am checking if length grew to detect if
set accepted element or if it is repeated
    if(len(not_repeated) != current_length + 1):
        repeated.add(x)
```

```
#Checking if length of non repeated elements is the same as original array
if(len(not_repeated) == len(array)):
    print("\nThere are no repeated elements in this Array")
else:
    print("\nThe repeated values in the Array are:\n" ,list(repeated))
```

Question 10:

```
#nums = [1,2,55,3,13,5,6,30,8,14]
#nums = [5,5,5,5,5,5]
nums = [0,-2,-3,-1,-55,-9]

print(nums ,":\n")

def find_largest(list):
    #Largest element is the only element
    if len(list) == 1:
        return list[0]

    else:
        current_largest = find_largest(list[1:])
        if current_largest > list[0]:
            return current_largest

        else:
            return list[0]

print("Largest Number in List: " ,find_largest(nums))
```

Question 11:

```
import math

def cosine(x, n):
    cos = 0
    for n in range(n):
        cos += ((-1)**n) * (x ** (2*n)) / (math.factorial(2 * n))
        # num ** x means num^x
    return cos

def sine(x,n):
```

```
    sin = 0

    for n in range(n):
        sin += ((-1)**n) / math.factorial(2*n+1)*(x**(2*n+1))
    return sin

print("Enter the order (n) for the expansion: ")
n = int(input())
#n = 1

print("Enter the value of x:")
x = float(input())
print("Cos(",x ,") to order",n ,"=",cosine(x,n),"\n")
print("Sin(",x ,") to order",n ,"=",sine(x,n))
```

Question 12:

```
fibonacci = [0,1]
sum = 0

print("Enter the value for n: ")
n = int(input())
error = "!! n cannot be equal to or less than 0 !!"


def Fibonacchi_Sequence(fibonacci,n,sum):
    for i in range(2,n):
        next = fibonacci[-1] + fibonacci[-2]
        fibonacci.append(next)
    print("The Fibonacci Sequence up to the first",n ,"elements is : \n"
,fibonacci ,"\n")

    for i in range(0,n):
        sum += fibonacci[i]

    print("The Fibonacci Sequence SUM of the first",n ,"elements is:",sum,"\n")

if(n <= 0):
    print(error)
elif(n == 1):
    fibonacci.remove(1)
    Fibonacchi_Sequence(fibonacci,n,sum)
else:
    Fibonacchi_Sequence(fibonacci,n,sum)
```