



SOEN 6441 - Advanced Programming Practices

CHEERS

Deliverable 1

Date: March 20, 2023

Submitted to: Prof. Pankaj Kamthan

GitHub Repository: [CLICK HERE.](#)

Team G

Bharath Ummaneni
Saryu Vasishat
Velagapudi Sri Neha
Manaswini Yarlagadda
Srujitha Yeruva

Contents

1	System Description	3
1.1	Introduction	3
1.2	Goals	3
1.3	Methodology	3
1.4	Constraints	4
1.5	Challenges	4
2	Solution 1: CRC Card Model	5
2.1	Roles	6
2.2	Responsibilities	6
2.2.1	Collaboration	7
3	Solution 2: Pseudocodes	8
3.1	Pseudocode for factorial function	8
3.2	Pseudocode for Pi function	8
3.3	Pseudocode for sine	8
3.4	Pseudocode for cosine	9
3.5	Pseudocode for Bisect function	9
3.6	Algorithms used	9
4	Solution 3: Incarnation 1 and 2	11
4.1	Incarnation 1	14
4.2	Incarnation 2	14
4.2.1	Code is IDE independent	16
4.2.2	Distributed version control system	16
4.2.3	Debugger	17
4.2.4	Object Oriented Principles	17
4.2.5	Exception handling	18
4.2.6	Error Messages	18
4.2.7	Assumptions	19
4.2.8	Instructions to process the source code	20

Abstract

A unique and challenging computational geometry problem CHEERS objective is to determine the distance required to align two circular coasters in such a way that the overlapping region's area is half of any one coaster's area. The solution must be obtained without utilizing pre-existing trigonometric functions or mathematical constants, such as pi and also it can't use any external programming tools except basic input, output, and arithmetic operations. To accomplish this, the program uses some formulas to calculate from scratch. To accomplish the project's goals the work is divided into several interrelated problems such as development of an accurate algorithm to compute alpha and l, error handling and efficient user input validation. Overall, the project enables us to understand the complexities of computational geometry and their real-world applications.

1 System Description

1.1 Introduction

The primary goal of the CHEERS project is to solve a practical problem in geometry by calculating the distance to align two circular coasters in such a way that the overlapping region's area equals half of any one coaster's area. accurate, and user-friendly program that can efficiently solve the problem of computing is implemented the distance required to move two circular coasters by taking the radius of any one of the coasters as input and then computes the angle α and the length of the segment X1X2. Finally the program displays the computed length to the user. CHEERS project applications can be found in many domains like manufacturing, engineering or design.

1.2 Goals

- To determine the distance two coasters have to be moved such that overlapping region is half the area of either coaster.
- To develop an efficient and accurate algorithm to compute required distance l.
- To document the methodology, incarnations, and insights of the project in accurate manner.
- The incorporation of OOP principles and the use of a CRC card model.

1.3 Methodology

CHEERS project involves the implementation of the equations for computing the angle α and the length of the segment X1X2 based on the given input parameters.

These are the steps involved

Input Validation: Program begins by validating the user input for the radius of the coasters. It checks that the input is a positive numerical value or it will throw an error message.

Solving the Equation $\alpha - \sin(\alpha) = \pi/2$: The next step involves solving the equation $\alpha - \sin(\alpha) = \pi/2$ to obtain the value of α . This equation cannot be solved

analytically and requires the use of numerical methods, such as the Bisection method.

Calculation of the length of the segment X1X2: Once the value of α is known, the program calculates the length of the segment X1X2 using the equation $l = 2R(1 - \cos(\alpha/2))$. This equation is a direct result of the overlap condition described in the problem statement.

Displaying the Result: Finally, the program displays the calculated length l to the user. If any errors occur during the computation, an error message is displayed.

1.4 Constraints

- The program will only accept numerical inputs for the coaster's radius
- The program will assume same units of measurement for the inputs and outputs.
- The program will not consider the coaster's physical properties, such as mass.
- The program will only display the length of the segment X1X2 as output but will not provide any graphical feedback of the same.
- The program is limited to only compute specific area of overlap. It cannot be used for other types of areas.
- The program must require specific operating system or IDE to run

1.5 Challenges

- To ensure the program's calculations are accurate for a wide range of input values
- Handling errors or exceptions that might occur during input validations , such as divide-by-zero errors or invalid inputs.
- To minimize any unnecessary computational overhead.
- To test the program thoroughly to ensure correctness for a wide range of inputs and scenarios, including edge cases and unexpected behavior

2 Solution 1: CRC Card Model

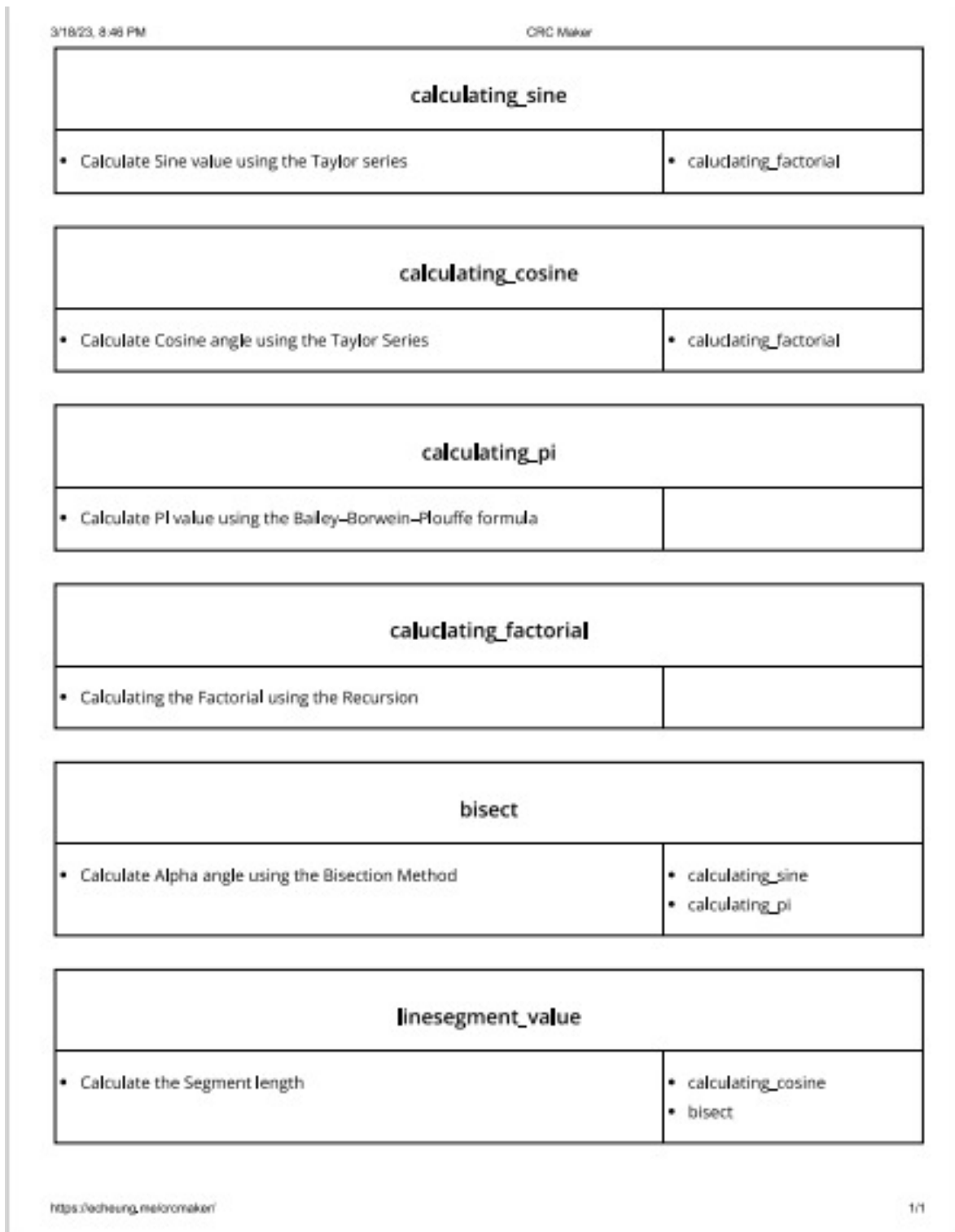


Figure 1: CRC Card Model

2.1 Roles

For the successful implementation of the project, our team decided to go with six cards for clear management and understanding of the code.

- "Calculation_Cosine" is used to calculate the Cosine angle.
- "Calculation_Sine" is used to calculate the Sin angle.
- "Calculating_pi" is used to find the value of pi. Our team is using the Bailey-Borwein-Plouffe formula from scratch.
- "Calculating_factorial" is using a recursion to find the desired output.
- "Bisect" CRC card is used to calculate the value of Alpha angle with the help of Bisection Method.
- "Linesegment_Value" is the main method which helps in determining the final output/length of our code.

2.2 Responsibilities

All above mentioned CRC functionalities are dependent on each other for final implementation.

- The "Caculating_Cosine" function is responsible for calculating the value of Cosine with the help of Taylor Series. In this series degree * $\pi / 180$ radians formula was used to calculate radians. Later the value of radians was substituted in the Taylor series for cosine(x).

$$\begin{aligned}\cos x &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}\end{aligned}$$

- Similarly, “Calculating_Sine” function was used for calculating the value of Sin.

$$\begin{aligned}\sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots \\ &= \sum_{n=1}^{\infty} (-1)^{(n-1)} \frac{x^{2n-1}}{(2n-1)!} \equiv \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}\end{aligned}$$

- The “Calculating_pi” function helps in determining the value of pi with the help of BBP (Bailey-Borwein-Plouffe) formula. The BBP formula is a remarkable option for calculating hexadecimal digits of π .
- The “Calculating_factorial” function is using recursion to calculate the factorial answer.

$$n! = n(n-1)(n-2) \cdots (2)(1)$$

- The “Bisect” function is solely responsible for finding the answer to the Alpha angle. Both “Calculating_sine” and “Calculating_pi” are used by the “Bisect” function for calculating the final result.
- The “Linesegment_value” uses both “Calculating_Cosine” and “Bisect” function to calculate the length of the required segment.

2.2.1 Collaboration

- It was mandatory for our team not to use predefined library functions like Sine, Cosine. So, our CRC cards are interdependent on each other for mathematical calculations.
- “Calculation_Sine”, “Calculating_Cosine” need the value/output of “Calculating_factorial” in Taylor series for calculating the final Sine and Cosine values. These both functions initialize factorial function before starting collaboration.

- “Bisect”, “Calculation_sine”, and “Calculating_pi” collaborate with each other to find the value of the Alpha angle. Later this angle is used by “Linesegment_value” to derive the value of l.

3 Solution 2: Pseudocodes

3.1 Pseudocode for factorial function

Define function calculatingfactorial (numbervalue):

 If (if numbervalue less than 1) :

 return 1

 else:

 return numbervalue * calculatingfactorial(numbervalue-1)

3.2 Pseudocode for Pi function

Define function calculatingpi(numbervalue):

 set pivalue to Decimal(0):

 set i to 0

 while (i less than numbervalue):

 pivalue += (Decimal(1)/(16**i))*((Decimal(4)/(8*i+1)) - (Decimal(2)/(8*i+4))-(Decimal(1)/(8*i+6)))

 i = i+1

 return pivalue

3.3 Pseudocode for sine

Define function calculatingsine(thetavalue):

 initialize xvalue to thetavalue

 m=0

 for k in range(0,10,1):

 y=(((-1)**k)*(xvalue)**(1+(2*k)))/(calculatingfactorial(1(2*k)))

 m+=y

 return m

3.4 Pseudocode for cosine

Define function calculatingcosine(thetavalue):

```
    initialize xvalue to thetavalue
    m=0
    for k in range(0,10,1):
        y=((-1)**k)*(xvalue**(2*k))/caluclatingfactorial(2*k)
        m+=y
    for j in range(1,25):
        pvalue=calculatingpi(24)
    return m
```

3.5 Pseudocode for Bisect function

Define function bisect(func, low, high):

```
    assertnot samesign(func(low), func(high))
    for i in range(10):
        set midpoint to (low + high) / 2.0
    if samesign(func(low), func(midpoint)):
        set low to midpoint
    else:
        set high to midpoint
    return midpoint
```

3.6 Algorithms used

Bailey–Borwein–Plouffe (BBP formula) is used to calculate PI value from Scratch

- **Speed:** BBP formula is best suitable for high performance applications as it computes the value of pi at faster pace.
- **Efficiency:** Since BBP formula requires only small amount of memory and computational resources it is highly efficient.

- Accuracy: The BBP formula individually calculates each hexadecimal digit of π and hence the errors do not propagate to subsequent digits.
- Portability: The BBP formula is highly portable as it can be implemented on any computing platform.
- Scalability: The BBP formula facilitates computation on multiple clusters and hence it is highly scalable.

The Taylor Series is used to calculate sine and cosine values from Scratch

- Accuracy: The Taylor series accurately calculates approximation of sine and cosine functions specifically when there are more terms in the series.
- Flexibility: The Taylor Series can be used to approximate the values of the sine and cosine functions at any point x , not just at specific values, which makes it a versatile tool.
- Efficiency: Since Taylor series can be used to compute values of cosine and sine at any point which requires only small amount of memory and computational resources it is highly efficient.
- Portability: Taylor series is highly portable as it can be implemented on any computing platform.

Bisection Method is used to calculate Root of equation from Scratch

- Simplicity: The Bisection Method does not involve complex mechanisms and hence it is relatively simple method
- Portability: Bisection method is highly portable as it can be implemented on any computing platform.
- Robustness: The Bisection Method is a reliable method to find the roots of equation and hence it is highly robust in nature.
- Accuracy: The Bisection Method gives accurate answers when interval is small, and number of iterations is large.

- Flexibility: Bisection Method can be used to find multiple roots of an equation and hence it is highly flexible.

Recursion is used to calculate Factorial from Scratch

- Simplicity: Recursion can provide a simple and elegant solution to calculating the factorial of a number, as the recursive function follows the natural definition of the factorial.
- Accuracy: The recursion Method gives accurate answers which makes it highly accurate.
- Flexibility: Recursion can be used to execute more complex cases and hence it is highly flexible in nature.
- Efficiency: Since tail recursion is used to calculate the factorial it is highly efficient

4 Solution 3: Incarnation 1 and 2

These are the steps taken to ensure that our code is modifiable, readable, reusable, testable, and understandable.

Modifiable

- Code was broken into smaller, reusable modules using functions and classes. This makes it easier to modify and test the code, as changes made to one module will not affect other modules. This approach also makes the code more readable, as it is easier to understand the purpose of smaller modules compared to larger ones.
- version control tools such as Git was used to track changes to the code and revert to previous versions if necessary. This helps to keep track of the code changes over time and maintain the code more efficiently.

Readable

- Descriptive variable names were used to clearly convey the purpose of the variable. Single letter variable names or abbreviations were avoided so that it would be easy understood easily.

- Blank lines were added between functions, classes, and blocks of code. This makes the code more readable and easier to understand.

Reusable

- There is no repetition of code. Instead, reusable functions or modules were created that can be called from various parts of the code. This approach reduces code duplication and improves maintainability, as changes to a single function can update the behavior in multiple places.
- Code is not specific to a particular use case. It can be easily adapted and reused.

Testable

- To verify the correctness and functionality of the given code, test cases were added which provides input to the code and verifies the output produced. This approach allows us to systematically test the code under various conditions and ensure that it meets the required specifications. By creating and running test cases, errors can be detected easily.

Understandable

- Comments were added to code to provide clear explanation of code. This approach can save time and effort in the long run, as it makes the code more readable and easier to maintain. Additionally, comments can provide useful context and insights into the thought process of the original developer, which can be valuable for future modifications or updates to the code.

These are the steps taken to ensure our code is general, robust, and usable.

General

- To create more versatile and reusable code, essential functionality was developed that can be applied to many use cases.
- Code was documented by providing clear explanation.

Robust

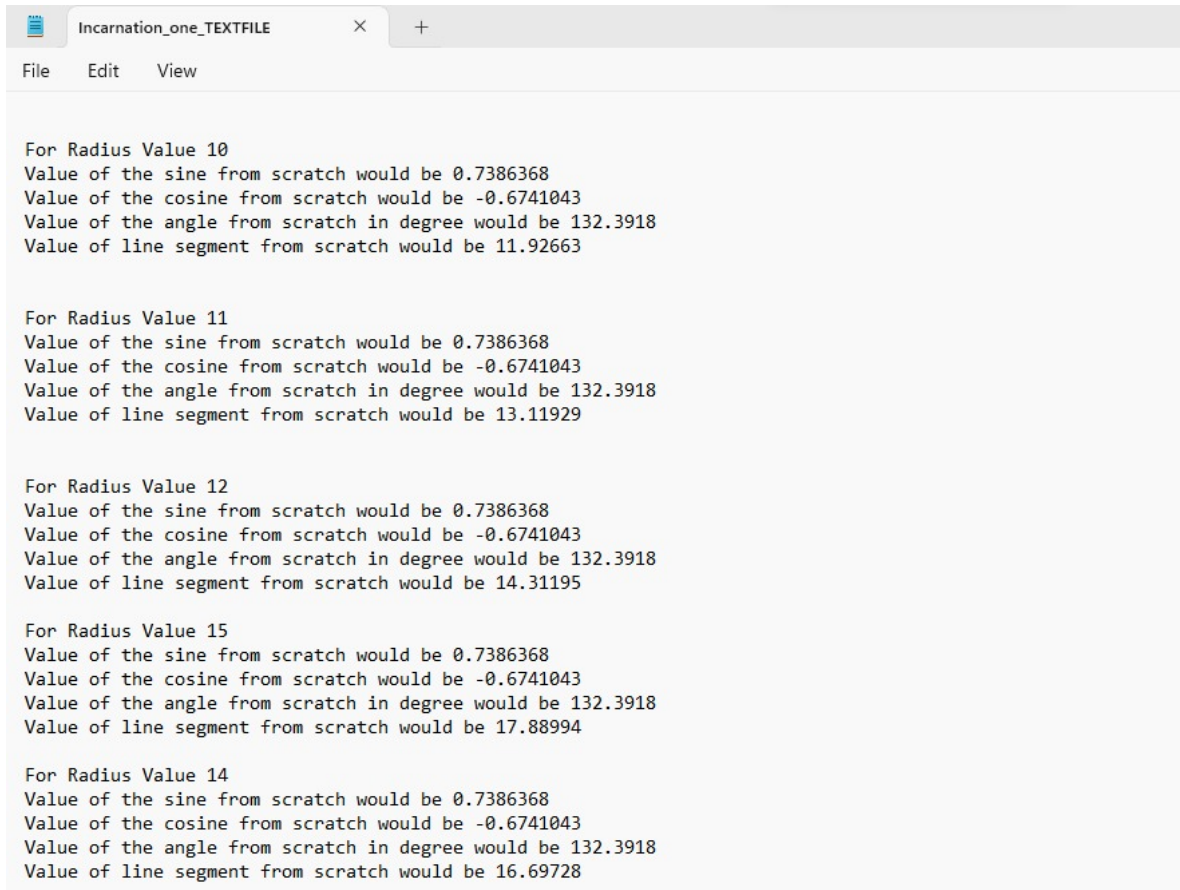
- To ensure that code is robust and reliable, try-catch blocks were implemented to handle errors gracefully, preventing them from crashing the program. This approach makes the code more resilient and less likely to fail, even when unexpected errors occur.

Usable

- To ensure that code is easy to use and accessible, number of dependencies it has on other libraries or packages were minimized. By reducing the number of external dependencies, process of running and testing the code can be simplified, making it easier for others to use without requiring them to install additional software.
- This approach involves identifying essential dependencies and incorporating them only when necessary, while seeking alternative solutions or implementing our own functionality when possible. This ensures that code remains self-contained and easy to use, promoting accessibility and reducing the likelihood of errors caused by complex dependencies.

4.1 Incarnation 1

Sample Output



```
Incarnation_one_TEXTFILE
File Edit View

For Radius Value 10
Value of the sine from scratch would be 0.7386368
Value of the cosine from scratch would be -0.6741043
Value of the angle from scratch in degree would be 132.3918
Value of line segment from scratch would be 11.92663

For Radius Value 11
Value of the sine from scratch would be 0.7386368
Value of the cosine from scratch would be -0.6741043
Value of the angle from scratch in degree would be 132.3918
Value of line segment from scratch would be 13.11929

For Radius Value 12
Value of the sine from scratch would be 0.7386368
Value of the cosine from scratch would be -0.6741043
Value of the angle from scratch in degree would be 132.3918
Value of line segment from scratch would be 14.31195

For Radius Value 15
Value of the sine from scratch would be 0.7386368
Value of the cosine from scratch would be -0.6741043
Value of the angle from scratch in degree would be 132.3918
Value of line segment from scratch would be 17.88994

For Radius Value 14
Value of the sine from scratch would be 0.7386368
Value of the cosine from scratch would be -0.6741043
Value of the angle from scratch in degree would be 132.3918
Value of line segment from scratch would be 16.69728
```

Figure 2: Output in plain text

4.2 Incarnation 2

For incarnation 2, these are the libraries used:

1. Pdb
2. Xml.dom.minidom
3. Decimal
4. Math
5. ElementTree

Pdb: For debugging we have used the “pdb” which is Python Debugger. “PDB” is a python built-in module that allows developers to debug the python code interactively.

Xml.dom.minidom: This module is used for generating XML file for various values of radius.

Decimal: Decimal is used to set the precision of a decimal value to 7.

Math: Math library is used to compare the original sine and cosine values with our computed values.

ElementTree: ElementTree module is used for creating and parsing the XML data

Sample Output

```
▼<Output>
  <Radius>For Radius : 10</Radius>
  <Alpha>Value of Angle in DEGREE is: 132.38458430464138</Alpha>
  <Length>Value of LENGTH SEGMENT is : 11.92663240454665540823953052</Length>
  <Sin>Value of Sine is: 0.738636738081631</Sin>
  <Cosine>Value of Cosine is: -0.6741036783434193</Cosine>
</Output>
```

Figure 3: Output in XML DTD format

```
▼<Output>
  <Radius>For Radius : 11</Radius>
  <Alpha>Value of Angle in DEGREE is: 132.38458430464138</Alpha>
  <Length>Value of LENGTH SEGMENT is : 13.11929564500132094906348357</Length>
  <Sin>Value of Sine is: 0.738636738081631</Sin>
  <Cosine>Value of Cosine is: -0.6741036783434193</Cosine>
</Output>
```

Figure 4: Output in XML DTD format


```

▼<Output>
  <Radius>For Radius : 12</Radius>
  <Alpha>Value of Angle in DEGREE is: 132.38458430464138</Alpha>
  <Length>Value of LENGTH SEGMENT is : 14.31195888545598648988743662</Length>
  <Sin>Value of Sine is: 0.738636738081631</Sin>
  <Cosine>Value of Cosine is: -0.6741036783434193</Cosine>
</Output>

```

Figure 5: Output in XML DTD format

4.2.1 Code is IDE independent

The source code was written in python. Creating source code that is not dependent on a specific Integrated Development Environment is crucial to guarantee the code's portability, flexibility, collaboration, maintainability, and longevity. This can be achieved by adhering to programming best practices and standards while writing the code in a modular and reusable manner, making it easy to understand and maintain. Code was checked by running it in command prompt to test whether it is independent of IDE.

```

Command Prompt
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ubhar>cd C:\Users\ubhar\Desktop\APP\EM726ASH2202CLA667_Updated (2) (1)\CHEERS_PROJECT

C:\Users\ubhar\Desktop\APP\EM726ASH2202CLA667_Updated (2) (1)\CHEERS_PROJECT>python main.py
*****CALCULATING EVERYTHING FROM SCRATCH*****
Value of PI would be:
3.141592
Value of ANGLE in RADIANS would be
2.310546875
Value of ANGLE in DEGREES would be
132.3846
Value of sine would be:
0.7386368
Value of cosine would be:
-0.6741043
Enter legal value of the RADIUS:-10
Value of LINE SEGMENT l would be:
11.92663

```

Figure 6: Output in Command Prompt

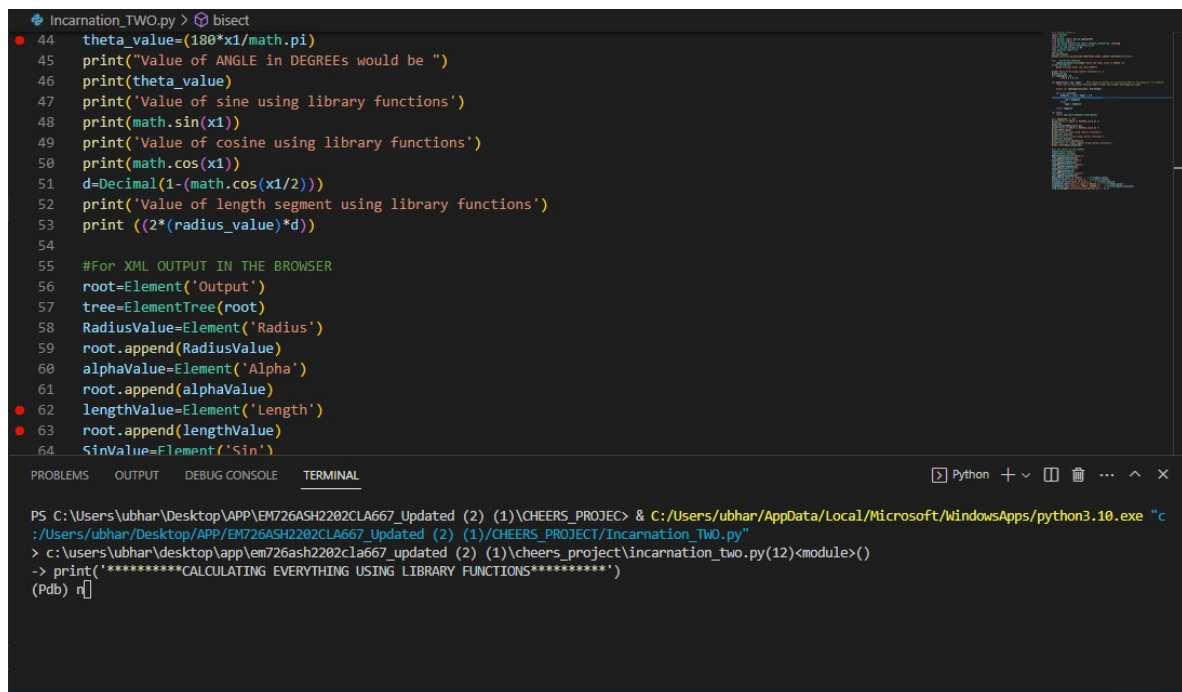
4.2.2 Distributed version control system

To manage code and dependencies, Git version control was used. This enables to move the code between different environments easily. Additionally, virtual environments were used to isolate code and its dependencies from the system Python installa-

tion. This helps to prevent conflicts between different versions of libraries and ensures that code runs consistently across different environments.

4.2.3 Debugger

The Python debugger allows to debug code by providing an interactive environment to examine the execution of program. It helps to identify and resolve errors in code. The Python debugger, which is also known as pdb, is part of the Python standard library and can be used from the command line or integrated into an IDE such as PyCharm or VS Code. To use the Python debugger, import pdb; pdb.set_trace() can be added to the code where debugging should be done. This line will cause the program to pause and enter the debugger's interactive mode. Breakpoints can be set through code where program can pause there when it entered to debugger's mode.



```
Incarnation_TWO.py > bisect
44  theta_value=(180*x1/math.pi)
45  print("Value of ANGLE in DEGREES would be ")
46  print(theta_value)
47  print('Value of sine using library functions')
48  print(math.sin(x1))
49  print('Value of cosine using library functions')
50  print(math.cos(x1))
51  d=Decimal(1-(math.cos(x1/2)))
52  print('Value of length segment using library functions')
53  print ((2*(radius_value)*d))
54
55  #For XML OUTPUT IN THE BROWSER
56  root=Element('Output')
57  tree=ElementTree(root)
58  RadiusValue=Element('Radius')
59  root.append(RadiusValue)
60  alphaValue=Element('Alpha')
61  root.append(alphaValue)
62  lengthValue=Element('Length')
63  root.append(lengthValue)
64  SinValue=Element('Sin')
```

```
PS C:\Users\ubhar\Desktop\APP\EM726ASH2202CLA667_Updated (2) (1)\CHEERS_PROJECT> & C:\Users\ubhar\AppData\Local\Microsoft\WindowsApps\python3.10.exe "c
:/Users/ubhar/Desktop/APP/EM726ASH2202CLA667_Updated (2) (1)/CHEERS_PROJECT/Incarnation_TWO.py"
> c:\users\ubhar\desktop\app\em726ash2202cla667_updated (2) (1)\cheers_project\incarnation_two.py(12)<module>()
-> print('*****CALCULATING EVERYTHING USING LIBRARY FUNCTIONS*****')
(Pdb) r[]
```

Figure 7: Proof of Debugger

4.2.4 Object Oriented Principles

Python is an object oriented programming language, it provides several features and principles that support object oriented programming concepts. OOP principles helps to

write more modular, maintainable, and scalable code. By using these principles, code will be easier to understand, debug, and extend.

```
class cheers :
    def calculating_pi(self,number_value):      #The Bailey-Borwein-Plouffe formula (BBP formula) for calculating PI value f
        pi_value = Decimal(0)
        i = 0
        while i < number_value:
            pi_value += (Decimal(1)/(16*i))*((Decimal(4)/(8*i+1))-(Decimal(2)/(8*i+4))-(Decimal(1)/(8*i+5))-(Decimal(1)/(8
            i += 1
        return pi_value
    def caluclating_factorial(self,number_value):  #RECURSION is used for calculating FACTORIAL from SCRATCH
        if number_value<1:
            return 1
        else:
            return number_value * self.caluclating_factorial(number_value-1)
```

Figure 8: The class cheers

4.2.5 Exception handling

In this project, exception was handled while taking the input. The exception handled is ValueError: The Python ValueError is an exception that occurs when a function receives an argument of the incorrect data type that prevents the system from crashing. The ValueError was also raised if the user is trying to enter the radius value less than zero.

```
while True :
    while True:
        try:
            radius_value = float(input("Please enter the radius of the circles: "))
            break
        except ValueError:
            print("Please enter a valid input")
    if radius_value < 0:
        print("Radius Can't be less than zero")
    else:
        break
```

Figure 9: Exception Handling

4.2.6 Error Messages

If the user gives a invalid input for the radius the error message will be displayed, the radius value should be a positive number. If the user enters a negative value or a value other than the numeric value the error message appears.

```

Please enter the radius of the circles: abc
Please enter a valid input
Please enter the radius of the circles: -10
Radius Can't be less than zero
Please enter the radius of the circles: c
Please enter a valid input

```

Figure 10: Output of Error Message

4.2.7 Assumptions

1. First decimal is set to 7 digits of precision.
2. Recursion is used in the calculation of the factorial, and the factorial later used in the computation of sine and cosine values.
3. Bailey Borwein Plouffe formula is used for calculating the pi value.

$$\pi = \sum_{k=0}^{\infty} \left[\frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) \right]$$

4. Taylor series of expansion formula is used for calculating the sine value.

$$\begin{aligned} \sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots \\ &= \sum_{n=1}^{\infty} (-1)^{(n-1)} \frac{x^{2n-1}}{(2n-1)!} \quad \text{or} \quad \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} \end{aligned}$$

5. Taylor series of expansion formula is used for calculating the cosine value.

$$\begin{aligned}\cos x &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}\end{aligned}$$

6. “Incarnation_one_TEXTFILE.txt” is used for storing the output values in the text file format.
7. “Incarnaton_two_XMLFILE.xml” is used for storing the output in an XML file.
8. “PDB” is used for debugging the python code. “PDB” is a python built-in module that allows developers to debug the python code interactively.

4.2.8 Instructions to process the source code

1. Visual code studio is used as the IDE.
2. For the Incarnation 1, Open the directory and run the file Incarnation_ONE. For the Incarnation 2, open the directory and then run the file Incarnation_TWO.
3. For the Incarnation_ONE the output which is in the structured plain text is stored in the Incarnation_one_TEXTFILE and for the Incarnation_TWO the xml output is stored in the Incarnation_two_XMLFILE.
4. Open the command prompt and set the directory using the cd command. Run the main.py file using the command python main.py.

Conclusion

In conclusion, the challenge of determining the distance required for two circular coasters to overlap by half their area has been successfully addressed. This involved

building the solution from scratch using Object-Oriented Design principles and without relying on external libraries or programming language support for mathematical functions. Through the outlined methodology, the correct distance can be accurately determined to achieve the desired overlap. This project underscores the significance of employing effective problem-solving techniques and the benefits of utilizing Object-Oriented Design.

References

- [1] Pankaj Kamthan. *Introduction to Domain Modeling*. 2023.
- [2] <https://www.thelearningpoint.net/home/mathematics/taylor-series-cosine-cos-expansion>
- [3] Adamchik, V. and Wagon, S. "A Simple Formula for pi." Amer. Math. Monthly 104, 852-855, 1997.
- [4] <https://observablehq.com/@rreusser/computing-with-the-bailey-borwein-plouffe-formula>
- [5] <https://www.sciencedirect.com/topics/computer-science/factorial-function>
- [6] @inproceedingsDevi2014AWF, title=A working Framework for the User-Centered Design Approach and a Survey of the available Methods, author=K. Rekha Devi and A. M. Sen and Kattamanchi Hemachandran, year=2014