

3PR3 - Practice Problems

Problem 1:

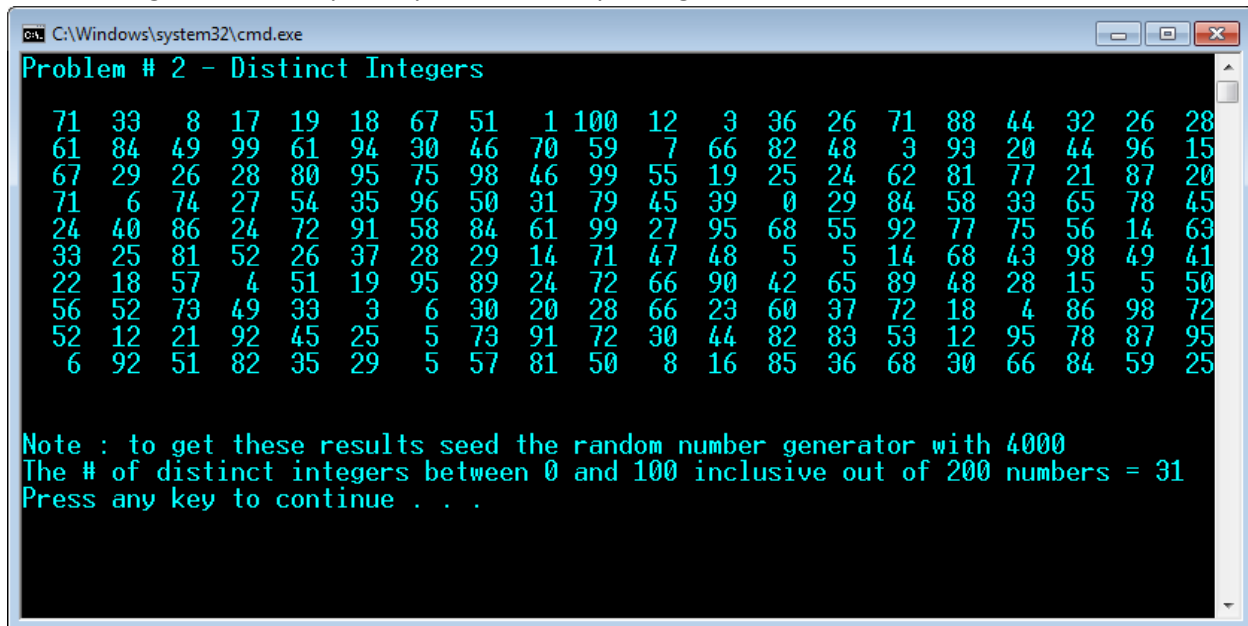
Write a function named **reduce** that takes two positive integer arguments, call them "*num*" and "*denom*", treats them as the numerator and denominator of a fraction, and reduces the fraction. That is to say, each of the two arguments will be modified by dividing it by the greatest common divisor of the two integers. The function should return the value 0 (to indicate failure to reduce) if either of the two arguments is zero or negative, and should return the value 1 otherwise. If the function returns 1, print the value of given fraction in reduced form within a main-function. [The greatest common divisor of two integers a and b is the largest positive integer that divides both numbers without a remainder.]

Problem 2:

Write a function named **distinct_numbers** that takes an array of random integers, numbers between 0 and 100, and counts the distinct integer values in the array. The function should take two arguments:

- an array of integers;
- an integer that tells the number of cells in the array.

The function should not return a value, but if duplicate integers are found, then the function should change the value of the argument that was passed to it so that the new value tells the number of distinct integers in the array. Test your function by calling it from main-function.



```
C:\Windows\system32\cmd.exe
Problem # 2 - Distinct Integers
71 33 8 17 19 18 67 51 1 100 12 3 36 26 71 88 44 32 26 28
61 84 49 99 61 94 30 46 70 59 7 66 82 48 3 93 20 44 96 15
67 29 26 28 80 95 75 98 46 99 55 19 25 24 62 81 77 21 87 20
71 6 74 27 54 35 96 50 31 79 45 39 0 29 84 58 33 65 78 45
24 40 86 24 72 91 58 84 61 99 27 95 68 55 92 77 75 56 14 63
33 25 81 52 26 37 28 29 14 71 47 48 5 5 14 68 43 98 49 41
22 18 57 4 51 19 95 89 24 72 66 90 42 65 89 48 28 15 5 50
56 52 73 49 33 3 6 30 20 28 66 23 60 37 72 18 4 86 98 72
52 12 21 92 45 25 5 73 91 72 30 44 82 83 53 12 95 78 87 95
6 92 51 82 35 29 5 57 81 50 8 16 85 36 68 30 66 84 59 25

Note : to get these results seed the random number generator with 4000
The # of distinct integers between 0 and 100 inclusive out of 200 numbers = 31
Press any key to continue . . .
```

Problem 3:

Define 3 structures to describe the following 3 geometrical entities:

1. A Point is defined by its x & y coordinates
2. A line is defined by its slope (m) & its intercept (c) using the equation $y=mx+c$
3. A rectangle is defined by the coordinates of 2 opposite corners; x1, y1, x2 & y2

After defining the 3 structures, develop the following functions:

1. A function named **line** that takes two point structures & calculates the line formed by them
2. A function named **intersection** that takes two line structures & finds the point of intersection between two lines.
3. A function named **in_out** that takes one point, one rectangle and finds whether the point is inside the rectangular or not.

In the main() develop a menu that asks the user to choose the execution of any of these 3 functions, then depending on the user choice, the user is prompted to enter the necessary values needed for each function. Test your functions by calling them from main().

Problem 4:

A complex number $x + jy$ can be described using a structure of 2 elements; real & imaginary parts. It can also be defined in a polar form by calculating both its magnitude & angle as follows:

$$mag = \sqrt{x^2 + y^2}$$

$$\text{If } (x, y) \text{ in Quadrant I:} \quad angle = \tan^{-1} \frac{y}{x}$$

$$\text{If } (x, y) \text{ in Quadrant II or III:} \quad angle = \pi + \tan^{-1} \frac{y}{x}$$

$$\text{If } (x, y) \text{ in Quadrant IV:} \quad angle = 2\pi + \tan^{-1} \frac{y}{x}$$

Define two different structures to represent a complex number; the first in Cartesian form & the second in polar form. Then develop multiple functions to perform the following complex calculations:

1. Generate random polar complex number
2. Conversion from Cartesian to polar form
3. Conversion from Polar to Cartesian form
4. Addition of 2 complex numbers
5. Subtraction of 2 complex numbers
6. Multiplication of 2 complex numbers
7. Division of 2 complex numbers
8. Quit

When the program runs, present the user with a menu with the above options, then interact with the user as necessary to gather values for complex numbers as well as displaying the result in an appropriate format.

Problem 5:

- (a) In math, a Cartesian plane is most commonly known as an X,Y graph. This graph has 4 quadrants. Here is the listing for the Quadrants and the relation to the (x,y) pair:

Quad. I: (x, y)	Quad. II: (-x, y)
Quad. III: (-x, -y)	Quad. IV: (x, -y)

For example, the pair (3, -5) lies in Quadrant IV and the pair (3, 3) lies in Quadrant I.

Generate 1000 random pairs, $-100 \leq x, y \leq 100$, and store them in a file **plane.txt**. Open this file for reading, and display (i) number of pairs in each quadrant but not on quadrant axes, a pair is on quadrant axes if $x = 0$, or $y = 0$ or $x = y = 0$ (ii) number of pairs on quadrant axes.

- (b) Generate a file of 3000 random dice throws, calculate how many doubles were rolled? It is known for this problem that each die is fair and are labeled with the numbers 1,2,3,4,5 and 6. It is also known that a double is defined as both dice showing the same number after they are rolled.

Problem 6:

Given the following number representing a pattern:

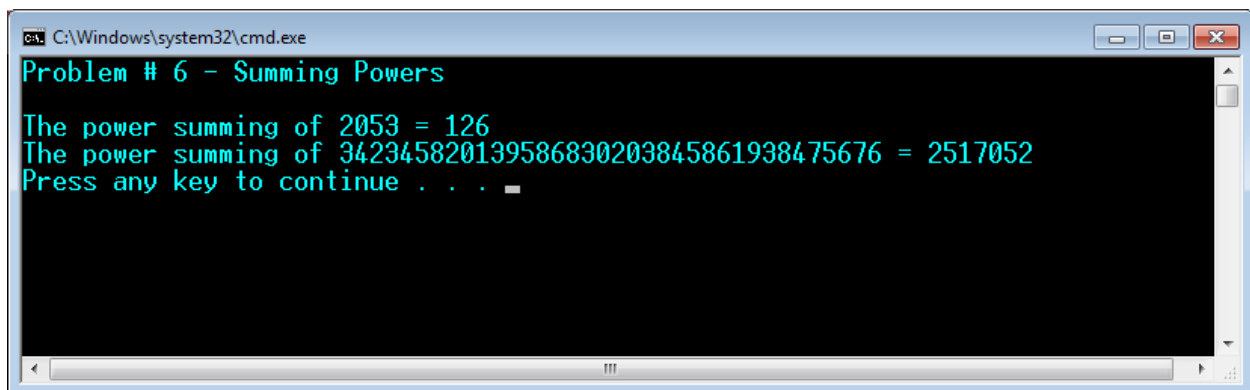
$$2053 = 2^0 + 5^3 = 1 + 125 = 126$$

Look above at the number "20"; in this sequence it represents 2 to the 0 power which is 1. The other number "53" represents 5 to the 3rd power which is 125. Then final answer is calculated by the sum of each term.

Given this number below:

342345820139586830203845861938475676

Write a program that will calculate the answer. Note that the number shown is too large to store in an integer number. You will need to use a char[].



```
C:\Windows\system32\cmd.exe
Problem # 6 - Summing Powers
The power summing of 2053 = 126
The power summing of 342345820139586830203845861938475676 = 2517052
Press any key to continue . . .
```

Problem 7:

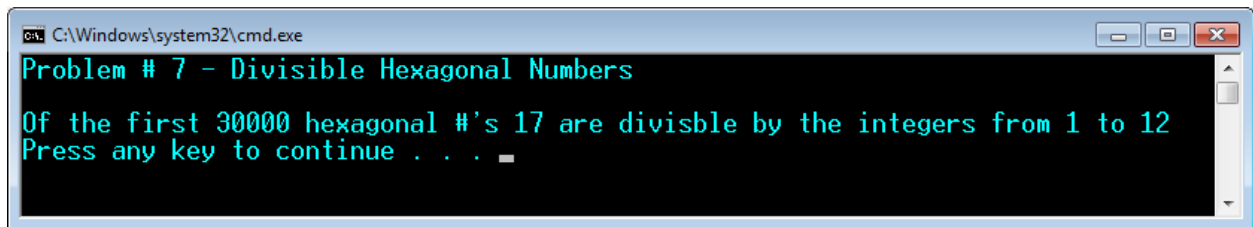
A hexagonal number is computed as follows using the formula:

$$H(n) = n(2n-1)$$

The first 5 hexagonal numbers are:

1 6 15 28 45

How many of the first 30000 hexagonal numbers are evenly divisible by all the numbers from 1 through 12?



```
C:\Windows\system32\cmd.exe
Problem # 7 - Divisible Hexagonal Numbers
Of the first 30000 hexagonal #'s 17 are divisble by the integers from 1 to 12
Press any key to continue . . .
```

Problem 8:

An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book. The first nine digits represent the book and the last digit is used to make sure the ISBN is correct. To verify an ISBN you calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third ... all the way until you add 1 times the last digit. If the final number leaves no remainder when divided by 11 the code is a valid ISBN.

For example 0201103311 is a valid ISBN, since $10*0 + 9*2 + 8*0 + 7*1 + 6*1 + 5*0 + 4*3 + 3*3 + 2*1 + 1*1 = 55$.

Each of the first nine digits can take a value between 0 and 9. Sometimes it is necessary to make the last digit equal to ten; this is done by writing the last digit as X. For example, 156881111X.

Write a function for validating ISBN code. This function should accept ISBN code and return true or false.

Problem 9:

The value of sine function can be calculated by McLaurin series as follows:

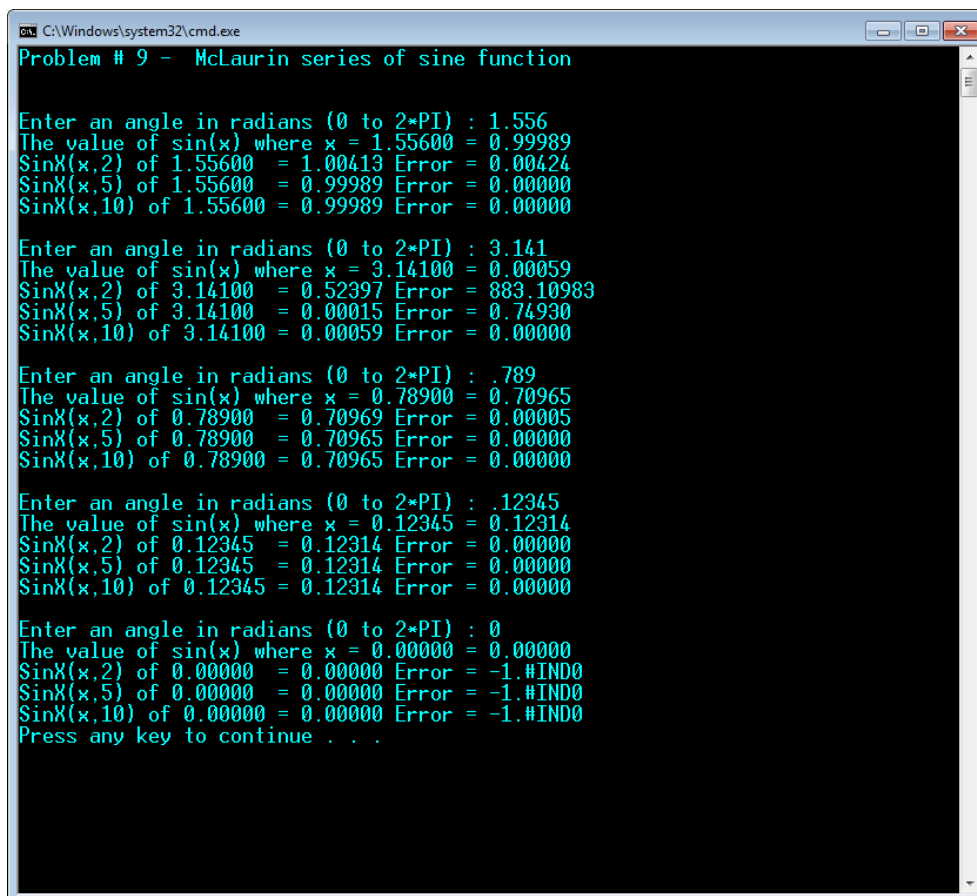
$$\sin(x) = \sum_{n=0}^k (-1)^n \frac{x^{2n+1}}{(2n+1)!},$$

where x is in radians and k represents the number of terms. We define the factorial of a non-negative integer n as, $n! = 1 \times 2 \times 3 \times \dots \times n$, with $0! = 1$. For example $4! = 1 \times 2 \times 3 \times 4 = 24$. The accuracy of $\sin(x)$ depends on the value of k.

- Write a function **sinX** that accepts the value of x and the number of terms as parameters. This function should calculate and return the value of $\sin(x)$ using McLaurin series.
- Write a function **factorial** that accepts an integer number as parameter and returns the factorial of a number as a double.

Ask the user to enter the value for x and the number of terms required in calculations. Use k = 2, 5, and 10 with every input value of x and display the following:

- Approximated value obtained from sinX function.
- Exact value obtained from sin function defined in <cmath>
- Display relative error (R.E.), where $R.E. = \left| \frac{Approximate - Exact}{Exact} \right|$



```
C:\Windows\system32\cmd.exe
Problem # 9 - McLaurin series of sine function

Enter an angle in radians (0 to 2*PI) : 1.556
The value of sin(x) where x = 1.55600 = 0.99989
SinX(x,2) of 1.55600 = 1.00413 Error = 0.00424
SinX(x,5) of 1.55600 = 0.99989 Error = 0.00000
SinX(x,10) of 1.55600 = 0.99989 Error = 0.00000

Enter an angle in radians (0 to 2*PI) : 3.141
The value of sin(x) where x = 3.14100 = 0.00059
SinX(x,2) of 3.14100 = 0.52397 Error = 883.10983
SinX(x,5) of 3.14100 = 0.00015 Error = 0.74930
SinX(x,10) of 3.14100 = 0.00059 Error = 0.00000

Enter an angle in radians (0 to 2*PI) : .789
The value of sin(x) where x = 0.78900 = 0.70965
SinX(x,2) of 0.78900 = 0.70969 Error = 0.00005
SinX(x,5) of 0.78900 = 0.70965 Error = 0.00000
SinX(x,10) of 0.78900 = 0.70965 Error = 0.00000

Enter an angle in radians (0 to 2*PI) : .12345
The value of sin(x) where x = 0.12345 = 0.12314
SinX(x,2) of 0.12345 = 0.12314 Error = 0.00000
SinX(x,5) of 0.12345 = 0.12314 Error = 0.00000
SinX(x,10) of 0.12345 = 0.12314 Error = 0.00000

Enter an angle in radians (0 to 2*PI) : 0
The value of sin(x) where x = 0.00000 = 0.00000
SinX(x,2) of 0.00000 = 0.00000 Error = -1.#IND0
SinX(x,5) of 0.00000 = 0.00000 Error = -1.#IND0
SinX(x,10) of 0.00000 = 0.00000 Error = -1.#IND0
Press any key to continue . . .
```

Problem 10:

Complete the following tasks related to file using C++ programming.

(a) Create a file "phoneLog.txt" with at least 500 records. This file should contain the following fields:

- *day number*, this represents the day number of the month and must be an integer value.
- *phone number*, this should contain 10 digits and must be stored in a character array.
- *call duration* (in minutes), this should be an integer value.
- *Status*, this represents the incoming or outgoing call, use 0 for incoming and 1 for outgoing.

Note: User input is not required, but you need to create this file using rand() function.

(b) Display a menu with the following options:

1. Display all records (this should display entire file in a tabular form with headings in first row)
2. Total number of minutes (this option should display the total minutes used in this month)
3. Distribution of minutes (this should display the total number of minutes for incoming and outgoing calls)
4. Total number of minutes used on a particular day (this option should ask the user to enter the day number and display the total number of minutes used on that day. For example, if user enters day = 10, this option should sum all minutes used on day = 10.)
5. Phone number starts with area code (this option should ask the user to enter the area code such as 905, 416, 289 and displays complete record of all phone number matching given area code.)

Test your program by first creating a file in (a) and then selecting menu options 1 through 5.

Note: Assuming number of records is unknown. All above options must be completed using functions.

Problem 11:

Write a program that creates two matrices A and B of dimension 4×4 with random integers, where an integer x satisfies $-9 \leq x \leq 9$.

- Write a function that prints a 4×4 matrix.
- Write a function **addition** that calculates the sum of two matrices
- Write a function **scalarMult** that multiply a scalar number with a matrix.
- Write a function **diagonal** that calculates the sum of diagonal elements.
- Write a function **transpose** that finds the transpose of a given matrix.
- Write a function **multiplication** that performs the multiplication of two matrices.

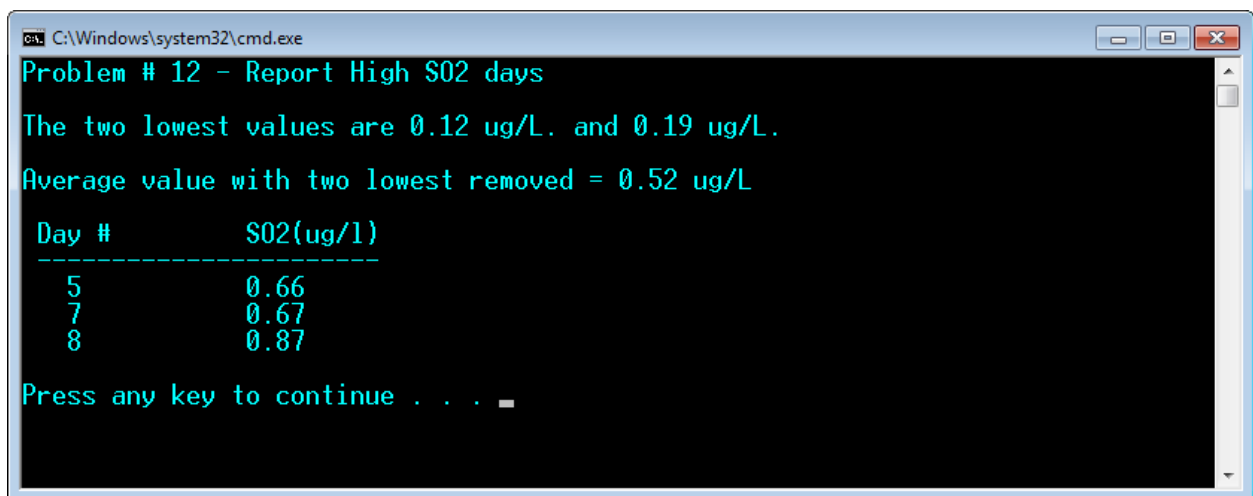
Test all function by calling them from main().

Problem 12:

Write a program that will analyze the following data that have been collected for SO₂ (ug/L) readings in ambient air at a site that is 50km away from a point source emission. The measurements have been collected at the same time each day over a 15-day period.

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SO2 (ug/L)	.45	.46	.12	.50	.66	.43	.67	.87	.41	.54	.19	.23	.56	.65	.35

Write a function called **avg2remove** that will calculate the average of all of the readings after eliminating the two lowest values. The function must return the average and the two lowest values. Print these values to the screen along with a formatted tabular output that reports the day # and value for all cases where the difference is greater than .13 ug/L from within the main function. All reported output should be displayed to 2 decimal places.



```
C:\Windows\system32\cmd.exe
Problem # 12 - Report High SO2 days
The two lowest values are 0.12 ug/L. and 0.19 ug/L.
Average value with two lowest removed = 0.52 ug/L
Day #      SO2(ug/l)
-----
5          0.66
7          0.67
8          0.87
Press any key to continue . . .
```