**Data Structures and Algorithms**

**Title: Algorithmic and Mathematical Functions**

**Lesson No: 2**

# LEARNING TASKS

## ENGAGE

Activity 1: Math is Amazing!

* 1. What is Pythagorean Theorem?

**The Pythagorean Theorem is a mathematical theorem invented by the mathematician Pythagoras pertaining that the square of the longest side of a right triangle (the hypotenuse) is equal to the sum of the squares of the other two sides.**

* 1. Design an algorithm that will look for the unknown side, given the two sides of a right triangle.

**1. Get the square of the 1st side.**

**2. Get the square of the 2nd side.**

**3. Get the sum of the squares of the two sides.**

**4.Get the square of the newly obtained sum, that will be the unknown side.**

* 1. Implement your algorithm by designing a program in C or Java.

public class PythagoreanTheorem

{

    static void Main(string[] args)

    {

        Console.WriteLine("Pythagorean Theorem Calculator\n");

        Console.Write("Enter the first side of your right triangle:\n-");

        double side1 = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter the second side of your right triangle:\n-");

        double side2 = Convert.ToDouble(Console.ReadLine());

        side1 = Math.Pow(side1, 2);

        side2 = Math.Pow(side2, 2);

        double sum = side1 + side2;

        double squareroot = Math.Sqrt(sum);

        Console.WriteLine("\nGot it! Here's how it goes: ");

        Console.WriteLine($"The square of both numbers are {side1} and {side2}.");

        Console.WriteLine($"{side1} + {side2} = {sum}.");

        Console.WriteLine($"The squareroot of {sum} is {squareroot}.\n");

        Console.WriteLine($"The answer is: {squareroot}");

    }

}

* 1. What is Quadratic Equation and when do you use it?

**The Quadratic Equation is basically a polynomial function with at least one squared term, which is why it’s called an equation of a second degree. The quadratic equation can prove to be useful when calculating the speed of something moving.**

* 1. Design an algorithm that will solve for unknown values using quadratic equations.

**1. Input the values of a, b, and c.**

**2. Perform the equation b^2 – 4 \* a \* c and store it in a variable.**

**3. Determine if there are two real solutions, one, or none out of the variable.**

**4. Print out the solutions (if any).**

* 1. Implement your algorithm by designing a program in C or Java.

class Program

{

    static void Main()

    {

        Console.WriteLine("Quadratic Equation Solver\n");

        Console.Write("Enter the coefficient a: ");

        double a = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter the coefficient b: ");

        double b = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter the coefficient c: ");

        double c = Convert.ToDouble(Console.ReadLine());

        double equation = b \* b - 4 \* a \* c;

        if (a == 0)

        {

            Console.WriteLine("This is not a quadratic equation.");

        }

        else if (equation > 0)

        {

            //for two solutions

            double x1 = (-b + Math.Sqrt(equation)) / (2 \* a);

            double x2 = (-b - Math.Sqrt(equation)) / (2 \* a);

            Console.WriteLine($"Two distinct real solutions: x1 = {x1}, x2 = {x2}");

        }

        else if (equation == 0)

        {

            //for one solution

            double x = -b / (2 \* a);

            Console.WriteLine($"One real solution: x = {x}");

        }

        else

        {

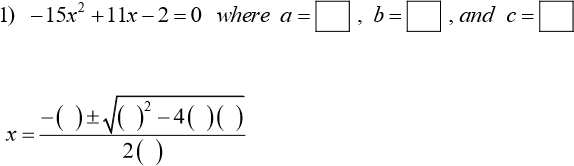
            Console.WriteLine("No real solutions.");

        }

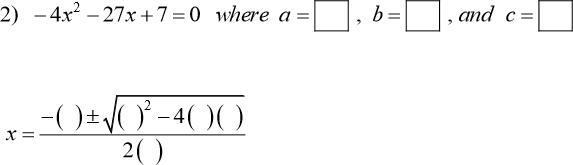
    }

}

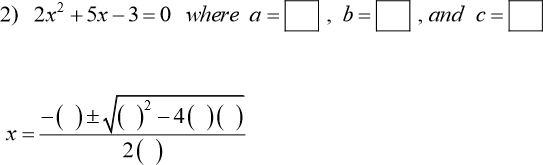
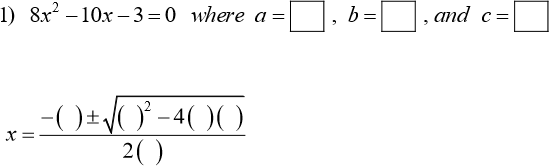
* 1. Test your program by solving the following:



a.



b.



c.

d.

Rubric:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Completed the activities and understood the topic based on the given  answer | Outstanding 50 points | Very Good 40 points | Good 30 points | Fair 20 points | No Work Output |

## EXPLORE & EXPLAIN

Design a program that you can use to solve the following:

public class MakeshiftCalculator

{

    static double num1;

    static double num2;

    static double output;

    public static List<double> CachedNums = new List<double>();

    public static List<double> CurrentNumber = new List<double>();

    public static double InputNum()

    {

        Console.WriteLine("-----------------------------------");

        Console.Write("Give me a number to work on: ");

        double yournum = Convert.ToDouble(Console.ReadLine());

        return yournum;

    }

    public static void ShowCached()

    {

        Console.WriteLine("-----------------------------------");

        foreach (var item in CachedNums)

        {

            Console.WriteLine(item);

        }

        Console.WriteLine("-----------------------------------");

    }

    public static void RetrieveCached()

    {

        Console.Clear();

        ShowCached();

        Console.Write("Choose a number to work with: ");

        double choose = Convert.ToDouble(Console.ReadLine());

        if (CachedNums.Contains(choose))

        {

            num1 = CachedNums.Find(n => choose == n);

        }

        else

        {

            RetrieveCached();

        }

    }

    static void Main(string[] args)

    {

        Console.WriteLine("-----------------------------------");

        Console.WriteLine("Hi, I'm a Makeshift Calculator!");

        CurrentNumber.Add(InputNum());

        Console.Clear();

        Console.WriteLine("Understood!");

        bool Ongoing = true;

        while (Ongoing)

        {

            Console.Clear();

            num1 = CurrentNumber[0];

            Console.WriteLine("-----------------------------------");

            Console.WriteLine($"Current Number: {num1}");

            Console.WriteLine("-----------------------------------");

            Console.WriteLine("+: Add\n-: Subtract\n\*: Multiply\n/: Divide\n^: Exponent\nsqrt: Squareroot\nstore: Store a number for later use.\nget: Retrieve a stored number.\n");

            Console.WriteLine("-----------------------------------");

            Console.Write("What do you wanna do?: ");

            string? input = Console.ReadLine();

            switch (input)

            {

                case "+":

                    num2 = InputNum();

                    output = num1 + num2;

                    CurrentNumber[0] = output;

                    continue;

                case "-":

                    num2 = InputNum();

                    output = num1 - num2;

                    CurrentNumber[0] = output;

                    continue;

                case "\*":

                    num2 = InputNum();

                    output = num1 \* num2;

                    CurrentNumber[0] = output;

                    continue;

                case "/":

                    num2 = InputNum();

                    output = num1 / num2;

                    CurrentNumber[0] = output;

                    continue;

                case "^":

                    num2 = InputNum();

                    output = Math.Pow(num1, num2);

                    CurrentNumber[0] = output;

                    continue;

                case "sqrt":

                    output = MathF.Sqrt((float)num1);

                    CurrentNumber[0] = output;

                    continue;

                case "store":

                    CachedNums.Add(num1);

                    Console.Clear();

                    Console.WriteLine("Your number has been stored.");

                    CurrentNumber[0] = InputNum();

                    continue;

                case "get":

                    RetrieveCached();

                    CurrentNumber[0] = num1;

                    continue;

                default:

                    Console.WriteLine("Enter a valid syntax!");

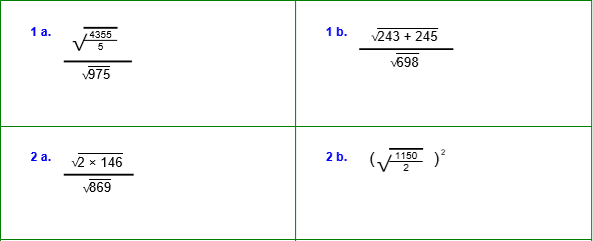
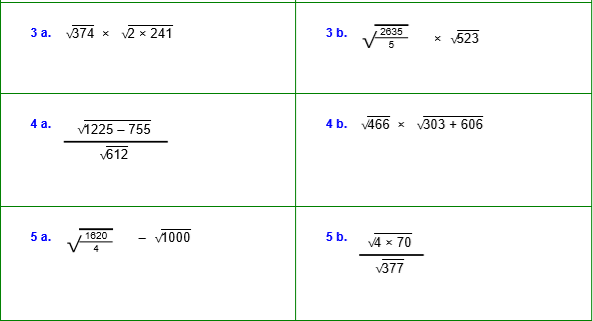
                    continue;

            }

        }

    }

}



1a. 0.95

1b. 0.84

2a. 0.58

2b. 575.0404

3a. 424.58

3b 525

4a. 0.88

4b. 650.84

5a. -11.5

5b. 0.86

Rubric:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Each correct | Question 1a | Question 2a | Question 3a | Question 4a | Question 5a | Total Score |
| answer will be given 3 points.  Total score  = 30 points |  |  |  |  |  |  |
| Question 1b | Question 2b | Question 3b | Question 4b | Question 5b |
|  |  |  |  |  |

## ELABORATE & EVALUATION

I. Answer the following question by encircling the letter of your choice or submit your answer on a separate sheet of paper. (20 points)

1. Two main measures for the efficiency of an algorithm are
   1. Processor and memory b. Complexity and capacity

c. Time and space d. Data and space

1. The time factor when determining the efficiency of algorithm is measured by
   1. Counting microseconds b. Counting the number of key operations

c. Counting the number of statements d. Counting the kilobytes of algorithm

1. The space factor when determining the efficiency of algorithm is measured by
   1. Counting the maximum memory needed by the algorithm
   2. Counting the minimum memory needed by the algorithm
   3. Counting the average memory needed by the algorithm
   4. Counting the maximum disk space needed by the algorithm
2. Which of the following case does not exist in complexity theory

a. Best case b. Worst case c. Average case d. Null case

1. The complexity of the average case of an algorithm is
   1. Much more complicated to analyze than that of worst case
   2. Much more simpler to analyze than that of worst case
   3. Sometimes more complicated and some other times simpler than that of worst case
   4. None of the above
2. Big-Omega notation sets
   1. a precise time complexity; b. a lower bound;

c. an upper bound; d. an upper and lower bound;

e. an indefinite estimate

1. For analyzing an algorithm, which has better computing time?

a. O (100 Log N) b. O (N log N) c. O (N) d. O (N2) e. O (2N)

1. For a problem of size n, a solution of the worst time complexity would be
   * 1. (constant time) b. 2 to the n power (exponential) c. n (linear)

d. none is worst e. n squared (quadratic);

1. log2(100) is closest to

a. 1; b. 2; c. 6; d. 20; e. 100

1. log2(1000) is closest to

a. 1; b. 2; c. 5; d. 10; e. 100

# REFERENCES

* + Fox, C. (2018). Java Data Structures and Algorithms 1st Ed.
  + Miller B. & Ranum D. (2013). Problem Solving with Algorithms and Data Structures Release 3.0
  + Goodrich, M. & Tamassia, R. Data Structures and Algorithms in Java 4th Ed.
  + Wikipedia
  + Arya Ashish (2010). 10 Effective Ways to Become a Good Programmer

<https://dzone.com/articles/10-effective-ways-become-good>

* + <http://what-when-how.com/compiler-writing/algorithmic-notation-compiler-writing-part-1/>
  + <https://bisma.in/algorithm-and-its-characteristics/>
  + [https://beginnersbook.com/2018/10/ds-asymptotic-](https://beginnersbook.com/2018/10/ds-asymptotic-notation/#%3A~%3Atext%3DAsymptotic%20Notations%20are%20the%20expressions%2Cthe%20complexity%20of%20an%20algorithm.%26text%3DAverage%20Case%3A%20In%20which%20we%2Cbetween%20best%20and%20worst%20case) [notation/#:~:text=Asymptotic%20Notations%20are%20the%20expressions,the%20complexity%2](https://beginnersbook.com/2018/10/ds-asymptotic-notation/#%3A~%3Atext%3DAsymptotic%20Notations%20are%20the%20expressions%2Cthe%20complexity%20of%20an%20algorithm.%26text%3DAverage%20Case%3A%20In%20which%20we%2Cbetween%20best%20and%20worst%20case) [0of%20an%20algorithm.&text=Average%20Case%3A%20In%20which%20we,between%20best%](https://beginnersbook.com/2018/10/ds-asymptotic-notation/#%3A~%3Atext%3DAsymptotic%20Notations%20are%20the%20expressions%2Cthe%20complexity%20of%20an%20algorithm.%26text%3DAverage%20Case%3A%20In%20which%20we%2Cbetween%20best%20and%20worst%20case) [20and%20worst%20case](https://beginnersbook.com/2018/10/ds-asymptotic-notation/#%3A~%3Atext%3DAsymptotic%20Notations%20are%20the%20expressions%2Cthe%20complexity%20of%20an%20algorithm.%26text%3DAverage%20Case%3A%20In%20which%20we%2Cbetween%20best%20and%20worst%20case).