



A TECHNICAL MAGAZINE

CORETECHRA

DEPARTMENT OF INFORMATION TECHNOLOGY

Vol.7
Issue 1

MAY 2021

OUR MISSION

To produce technically competent and skilled engineers to meet the challenges of the IT industry

OUR VISION

- Establishing innovative teaching learning practices with competent faculty and state of the art facilities.
- Enriching knowledge on latest technological advancements through industrial collaborations.
- Developing moral and ethical values through extension activities.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1:

The graduates will have a successful career in the field Information Technology and related domains.

PEO 2:

The graduates will provide solutions by applying analytical skills for the real-world problems in IT Industry.

PEO 3:

The graduates will involve in lifelong learning and as part of team in multidisciplinary projects with ethical values.

PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO 1:

Identify, formulate and solve engineering problems by applying programming concepts and algorithmic principles in the field of IT.

PSO 2:

Analyze, design and develop Software applications, Networking and Data Management technologies for efficient IT based systems.

Table of Content

| | |
|---|----|
| AN ULTRAFAST LIGHT TIMESERIES STORAGE ENGINE | 01 |
| ASYNC/AWAIT | 06 |
| CUSTOMIZABLE WPF MESSAGEBOX | 10 |
| LOGICAL THINKING!! | 14 |
| HOW TO SEND INPUTS USING C# | 16 |
| STERN-BROCOT TREES AND THEIR APPLICATIONS | 21 |
| WHY SOFTWARE BUSINESSES FAIL | 26 |
| PROJECT GLIMPSES | 31 |
| LOGICAL THINKING! Solutions | 33 |

AN ULTRAFAST LIGHT TIMESERIES STORAGE ENGINE

A article ultrafast timeseries storage engine is introduced in this article. The solution provides a AP/ITnumber of examples showing the performance for various workloads by writing 1,000,000,000,000 timeseries values to the timeseries storage. A timeseries is a table that is indexed according to time, and many machine learning algorithms rely heavily on the data found in timeseries tables. Anything that changes over time is considered information in a timeseries, such as:

- ✓ Stock prices and volume.
- ✓ Precipitation.

You can make educated forecasts about future flooding or droughts, or the price of electricity if hydropower is the primary source of electricity, by analysing stock prices and precipitation, respectively. Leaving aside cat images, timeseries are likely the most common type of data to be maintained.

Ellen Friedman and Ted Dunning present a convincing case in their book Time Series Databases: New Ways to Store and Access Data for the use of a hybrid noSQL database to create a high-performance timeseries database engine utilising Open TSDB as a reference case. The concept is simple: Use blobs to store the timeseries data.

My intention for this article, and the accompanying source code, is to show that the concept works – and that it can be implemented in easily understood C++. You can easily follow the code for this article if you have a background as a C# developer, deep C++ knowledge is not required.



Building the Code:

The code cannot be created or executed without the boost C++ libraries. The included Visual Studio projects assume that the library files are located in \$(BOOST_ROOT) stagelib and that the environment variable BOOST_ROOT points to the directory where you unpacked the download.

Make sure to compile for x64, and adjust the Enable Enhanced Instruction Set setting under C/C++ Code Generation to a value appropriate for your system if you intend to test the code on a platform that does not support AVX2.

The full path to a directory where the tests can construct a subfolder for the database files must be set in the environment variable HCC_TEST_DATA_ROOT used by the performance tests.

Performance:

Since I refer to this as an ultrafast timeseries storage, I ought to be able to provide data to support that claim. As a result, before diving into the specifics of the solution, I'd want to share the outcomes of some performance tests I ran to gauge how well this solution performs. When analysing the outcomes, keep in mind that the programmes' source code is included in the download for this article: Please bear in mind that these tests weren't conducted on a high-performance server, but rather a laptop.



The timings cover the interval from before the first timeseries point is written to when the last one is written and the transaction is committed to the database file. A new transaction scope is used for reading.

Database directory:F:\Database\LMDB

Wrote 1000000000 timeseries points in 43.384699 seconds - points pr. second: 23049600.966461

Read 1000000000 timeseries points in 1.232314 seconds - points pr. second: 811481423.445532

Sum inserted:500000000067108992.000000,

Sum read: 500000000067108992.000000

This is undoubtedly encouraging because the programme.

- ✓ More than 23 million timeseries points per second were added.
- ✓ Per second, read more than 811 million timeseries points.

There is enough information to show that the timeseries storage engine is more than just a play with one billion timeseries points. The timeseries point consists of three fields in a record:

- ✓ Timestamp: DateTime
- ✓ Flags: Int64
- ✓ Value: double

One billion timeseries points have a size of almost 24 Gb because each record is 24 bytes in size. Sum read is the total of all timeseries point values read from the storage, and Sum inserted is the total of all timeseries point values pushed to the storage. The same sums demonstrate that the data we receive back is the same as what was written. Run it once more, and the timeseries points are written even faster:

Database directory:F:\Database\LMDB

Wrote 1000000000 timeseries points in 23.693139 seconds - points pr. second: 42206310.574444

Read 1000000000 timeseries points in 1.181736 seconds - points pr. second: 846212833.697684

Sum inserted:500000000067108992.000000,

Sum read: 500000000067108992.000000

Currently, the programme:

- ✓ more than 42 million timeseries points per second were added.
- ✓ per second, read more than 846 million timeseries points.

Because the timeseries storage engine reuses the storage file created during the initial run, the write performance increased. Although the performance appears to be good, this is not a typical application for a timeseries storage engine. I have examined numerous timeseries engine benchmarks over the years, and they often write timeseries points in batches. The timeseries points are written to the timeseries in batches of 250 values using HExPerf03.exe, which writes one billion timeseries values over 10,000 timeseries:

Reading the data and generating a straightforward checksum:

```
size_t totalRows = 0;
for ( auto& timeseriesId : timeseriesIds )
{
    timeseriesCursor2.ChangeTimeseries( timeseriesId );
    totalRows += timeseriesCursor2.ForEach( []( const Point& point, double& sumRead )
    {
        sumRead += point.Value( );
    }
    );
}
```



```
}, sumRead );  
}
```

Output:

Database directory:F:\Database\LMDB

Inserted 1000000000 timeseries points into 10000 timeseries in 27.847767 seconds

- points pr. second: 35909521.937612

Read 1000000000 timeseries points from 10000 timeseries in 2.563846 seconds

- points pr. second: 390039103.370308

Sum inserted:50000500000000.000000,

Sum read: 50000500000000.000000

The performance decreased as anticipated, but the timeseries engine can still

- ✓ More than 35 million timeseries points should be added every second.
- ✓ Per second, read more than 390 million timeseries points.

How well a timeseries storage engine handles writing one value at a time to the timeseries storage is the test that is most intriguing. If 10,000 sensors were reporting changes simultaneously, the workload would be substantially different from what has been shown thus far.

Database directory:F:\Database\LMDB

Inserted 1000000000 timeseries points into 10000 timeseries in 76.781371 seconds

- points pr. second: 13023992.431581

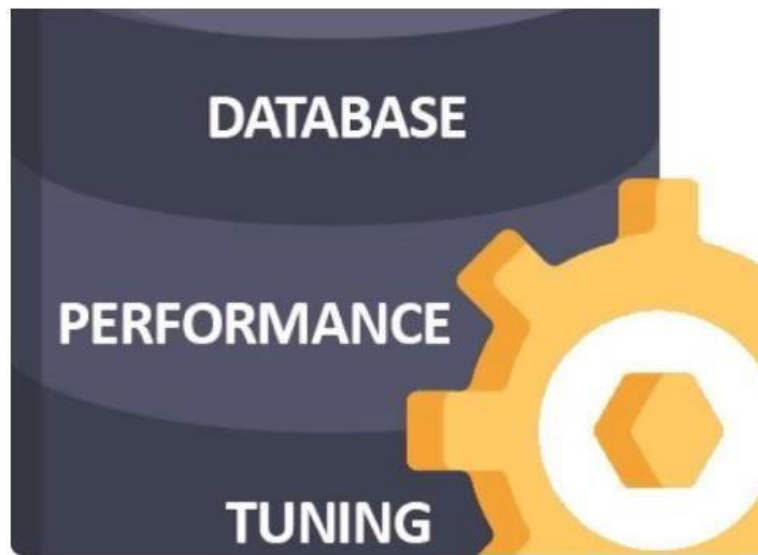
Read 1000000000 timeseries points from 10000 timeseries in 2.515501 seconds

- points pr. second: 397535075.811728

Once again, the program's performance declined.

- ✓ More than 13 million timeseries points per second were added.
- ✓ Per second, read more than 397 million timeseries points.

It should be noted that while the write performance is generally consistent, the read performance varies significantly and, for this workload, ranges between 300 and 400 million timeseries points per second. This is most likely a result of the read performance being strongly dependent on the cache mechanisms offered by Windows for memory mapped IO, which aims to balance the needs of all the apps that are currently executing on the system.



The cache is hot, which is the primary cause of the remarkably fast read speed because each of the aforementioned tests wrote all the data right before reading it back again.



Introduction:

Async/Await functions are explained in this documentation. It might be a challenging topic. Beginning with some straightforward fundamental concepts, we'll gradually proceed to more complex ones. Those who study best visually should benefit from the diagrams.

Synchronous (Sync) Method:

IA synchronous (sync) methods are normal methods that lack an await and are not async-marked. For instance:

```
private void Foo()
{
    ...
}
```

Asynchronous (Async) Method:

A method that is designated async and contains an await is referred to as an asynchronous (async) method. For instance:

```
private async Task FooAsync()
{
    await BarAsync();
}
```

Synchronous Call:

A call without an await is said to be synchronous. It might or might not give back a value. For instance:

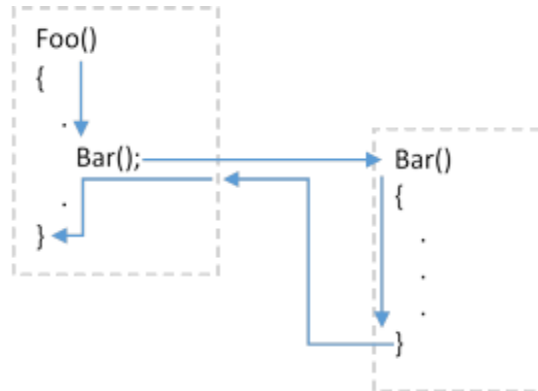
```
Bar();
OR
int x = Bar();
```

Asynchronous Call:

A call that uses await is said to be asynchronous. It might or might not give back a value. For instance:

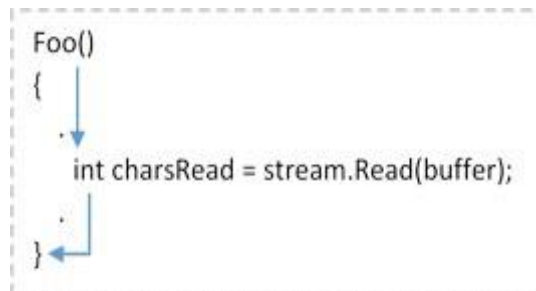
```
await BarAsync();
or
int x = await BarAsync();
```


Sync Calling Sync A Regular Call:



Foo() calls **Bar()**. **Bar()** runs and then returns to **Foo()**.

A Regular Call to `stream.Read()`:



`Foo()` calls `stream.Read()`. The thread waits until `stream.Read()` completes, and then continues.

Async awaiting Async `await stream.ReadAsync()`:



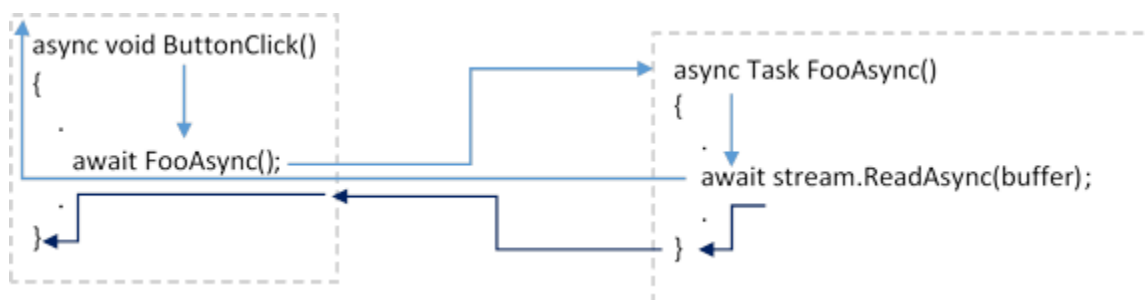
Using one thread to accomplish two tasks at once:

In the case mentioned above, await stream is called by ButtonClick(). ReadAsync(). The UI thread is given the opportunity to perform other tasks by returning to the caller of ButtonClick() before the read is finished. We may say that we are currently performing two tasks at once:

- ❖ We are anticipating the conclusion of the ReadAsync() function.
- ❖ Messages in the Message Queue are being processed by the UI thread.

Whether or not waiting actually qualifies as accomplishing something depends on semantics (typically it does not count). We are actually waiting, and everything is set up so that the remaining code will run after the await stream when this wait is finished. Although ReadAsync() will run, it is a passive wait, therefore the UI thread won't be blocked at this time. We also wait passively for the await stream while the UI thread is free to perform other tasks. ReadAsync() to finish. Keep in mind that the UI thread is the sole thread running and that it is still the only thread performing any work.

ButtonClick calling FooAsync calling stream.ReadAsync():

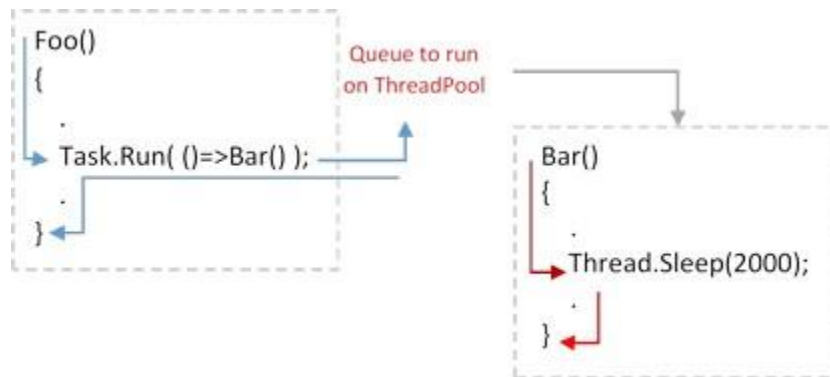


Calls to ButtonClick() wait for FooAsync (). await stream is called by FooAsync(). ReadAsync(). An unfinished Task is returned to the caller of FooAsync instead of waiting for the read to finish (). await When FooAsync() notices that the Task that was returned to it isn't complete, it calls back the UI thread's Message Loop. As a result, the screen can be updated and the UI thread can process further messages in the Message Queue.

Later, when ReadAsync() is finished, the remainder of FooAsync() is executed (dark blue). The remainder of ButtonClick() executes after FooAsync() sends a finished Task to await FooAsync().

Sync calling Task.Run():

Sync → Task.Run() → Sync:



1. Task. Run() pools a ThreadPool thread to execute Bar().
 2. Foo() waits till task t is finished.
- ❖ By changing its execution state to "WaitSleepJoin" (the blocked state), the thread that is currently running Foo() enters a wait state and surrender the remaining processor time slice. (Doing so allows the CPU to operate additional threads.) No processor time is used by the thread until its blocking condition is met.
 - ❖ When Bar() is finished, the thread that is now running Foo() has its execution status set back to "Running," and it resumes running once the Thread Manager has a time slice available for it.

It is not advisable to use.Wait() on the UI thread because doing so may cause the programme to become unusable. We don't want to keep the UI thread locked up while doing nothing. (Consider making Foo() an await Task and async function. instead of Run(()=>Bar()).)

We should avoid using. Wait() if, however, Foo() is being executed on a ThreadPool thread because we would be blocking that thread while we wait for another ThreadPool thread to execute Bar ().

GNANASEKARAN K
II-IT

The infinitesimally thin text that Windows uses unless you increase the font size in steps of 10% is a particular discomfort associated with deteriorating eyesight. The writing is WAY bigger than it has to be, even at 110 percent, which is the issue with this. In any case, this little text cannot be modified because it is used in the default WPF message box. Actually, there is NOTHING that can be altered regarding the default WPF message box. I was ultimately overcome by this realisation, and I was compelled to develop this code



First, though, complaints:

I was shocked and horrified to learn that the common WPF message box wasn't a WPF-specific object when I first started my investigation on it. It turns out that there isn't a single drop of WPF goodness in it; instead, it is totally implemented using the Win32.Interop feature. A Microsoft supporter would attempt to argue that their strategy likely decreased the overall. I'm not buying it. Net code footprint increased by a few thousand bytes. Here was a genuine opportunity, but Microsoft blew it. All uses Windows system properties, and that's it, so that's why it's not adjustable in any manner. You cannot deviate from such restrictions. A WORSE PROGRAMMER! VERY WRONG!

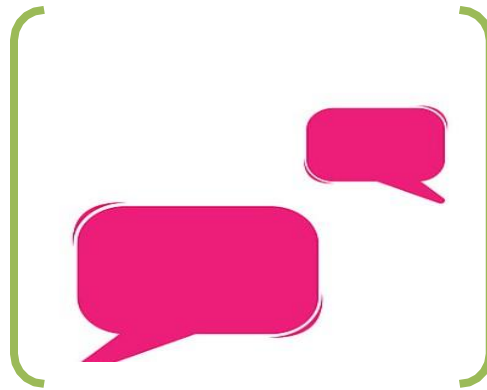
What I Ended Up Wanting to Do:

My main objective was to create a substitute for the conventional message box that was completely WPF-based, sufficiently flexible, and presented an interface that was as similar to the original message box as feasible. Without too much heartburn, I managed to do this using some static magic.

Emulating Default WPF Message Box Features:

The first thing I had to do was imitate the default WPF MessageBox. The Show() method was initially overloaded to accommodate all potential argument combinations (in earlier iterations of the code), but the sheer number of overloads was becoming unmanageable. Offering the ability to specify the default button was what truly sent me over the edge; it was just a bridge too far, so I made the following decision:

```
MessageBoxEx.Show(string msg, params object[] parameters)
```



The user is able to specify any other parameter he wants, as well as the message text. Yes, the developer can specify more than one of any parameter when using this tactic, but the approach has various safeguards against foolish developer gimmicks. Variables in the method are first set to sensible default values. A specific variable is set as the parameters are processed until the variable's value no longer equals the default value. I prefer not to stop processing unless something truly catastrophic has occurred, and letting the programmer to be an idiot isn't a catastrophe as much as triggering an exception whenever a certain variable is set more than once.

Extra Features:

In addition to the basic features, I additionally included support for changing the elements listed below. This is the main motivation behind why this code was created in the first place.

- ✚ Font Family
- ✚ Font Size
- ✚ Text and background colour in the message area.
- ✚ colour of the button panel backdrop.
- ✚ support for custom button templates.

- ✚ improved support for default buttons.
- ✚ Set the widest message box you can

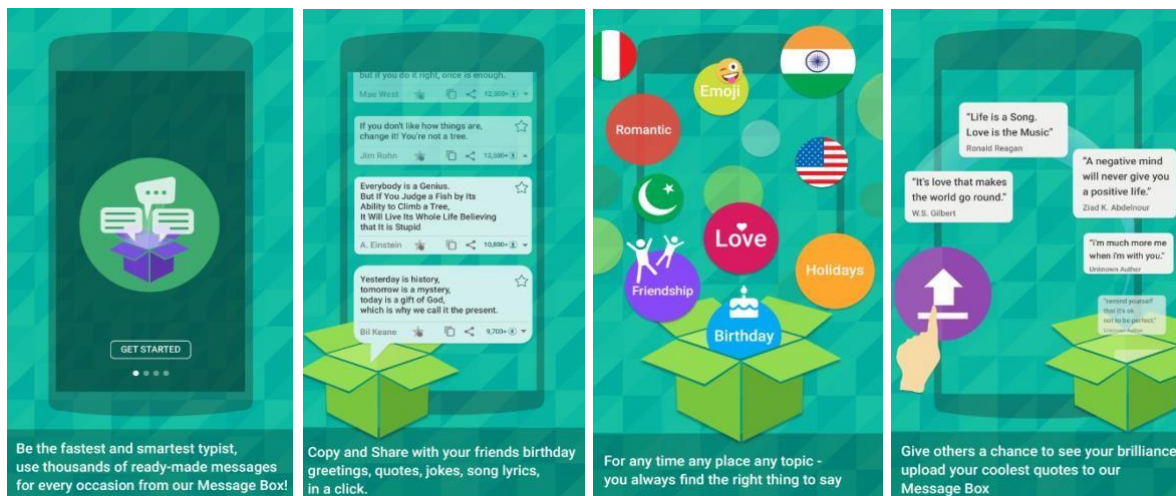
Notable Features Missing:

If you have ever developed software for Windows. You've definitely used Windows at some point, Forms. You might have observed that Microsoft chose not to support MessageBox Button when it came to MessageBox. MessageBox Button, Abort Retry Ignore, etc.

RetryCancel. This also means that certain buttons are not accessible when the button setting is set to the default. See the next article section for more information on how MessageBoxEx restores these buttons (and their respective defaults) to the game.

Extended Capability:

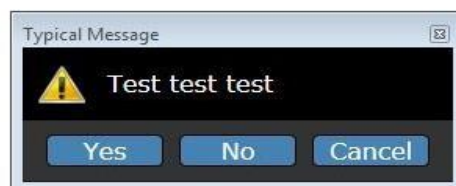
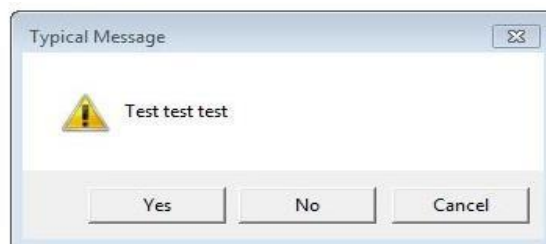
I made the decision to provide capabilities not found in the default WPF message box when I was designing the MessageBoxEx class. This comprised the previously mentioned missing buttons, a checkbox, a clickable URL, a clickable icon (to execute external code), and more default button choices. The Message Box Ex.Show Ex() method, which provides support for the Message Box Button Ex enumerator, and the MsgBox Extended Functionality class, which enables you to add the checkbox, url, and clickable icon, help distinguish the standard content from the expanded functionality.



The Sample Application:

The sample programme is straightforward in and of itself, offering a number of buttons designed to test the functionality of the message box. The screen photos that follow don't actually correspond with the buttons on the main window, but they do show a variety of message

length, icon, and button combinations as well as varied icon alignment scenarios. To obtain a more up to date sense of things, you should experiment with the example app. The usual MessageBox version is presented before each screenshot.



**MOHAMMED
ASHIK A
III-IT**

LOGICAL THINKING!!

Problem 1:

Write a Java program to check whether a given number is an ugly number.

In number system, ugly numbers are positive numbers whose only prime factors are 2, 3 or 5. First 10 ugly numbers are 1, 2, 3, 4, 5, 6, 8, 9, 10, 12. By convention, 1 is included.

Test Data: Input an integer number: 235

Sample Output:

Input an integer number: 235

It is not an ugly number.

Refer Page No:33

Problem 2:

Write a Java program to generate and show all Kaprekar numbers less than 1000.

In number theory, a Kaprekar number for a given base is a non-negative integer, the representation of whose square in that base can be split into two parts that add up to the original number again. For instance, 45 is a Kaprekar number, because $45^2 = 2025$ and $20+25=45$. The first few kaprekar numbers in base 10 are:

1, 9, 45, 55, 99, 297, 703, 999, 2223, 2728, 4879, 4950, 5050, 5292, ...

Refer Page No:33

Problem 3:

Write a Java program to find and print the first 10 happy numbers.

Happy number: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1, or it loops endlessly in a cycle which does not include 1.

Example: 19 is a happy number

$$1^2 + 9^2 = 82$$

$$8^2 + 2^2 = 68$$

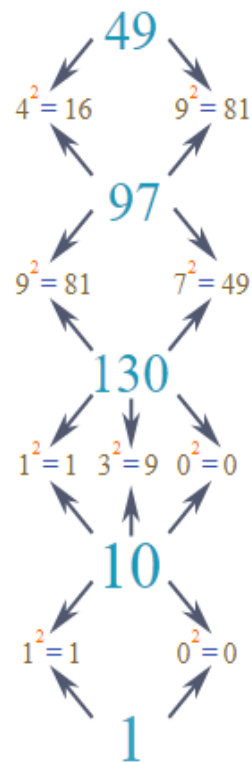
$$6^2 + 8^2 = 100$$

$$1^2 + 0^2 + 0^2 = 1$$

Happy Number

Example

- Pick a number
- Add result of the square of each digit
- Go same for new number
- If reach 1 this is Happy Number



A Happy Number

Refer Page No:34

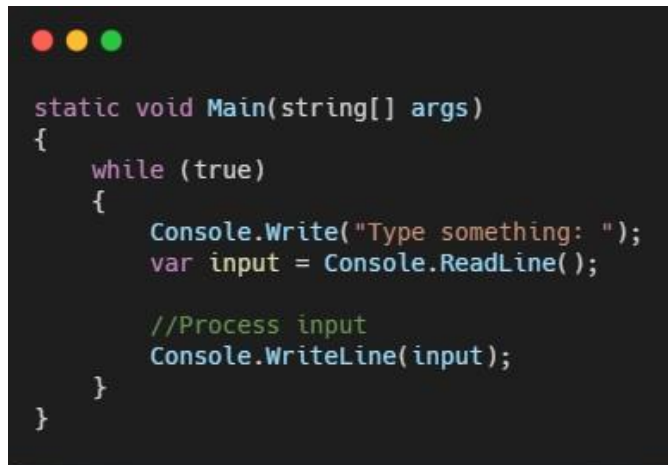
HOW TO SEND INPUTS USING C#

We will look at sending inputs using C# in this tutorial. You can play around with this or use it to automate chores.

With the code:

We'll be utilising user32.dll's SendInput method. We won't use keybd event and mouse event because they are out-of-date and might not function properly in future Windows versions.

An API called DirectInput allows users to provide input through input devices (keyboard, mouse, etc.). We can send hardware or virtual scan codes, depending on a few flags that we'll cover later. DirectInput may disregard virtual scan codes, which would prevent our inputs from being processed. Hardware scan codes are more akin to manually hitting a key.



```
static void Main(string[] args)
{
    while (true)
    {
        Console.Write("Type something: ");
        var input = Console.ReadLine();

        //Process input
        Console.WriteLine(input);
    }
}
```

The size of our INPUT struct, the number of inputs, and an array of INPUTs for the inputs we wish to transmit are the three parameters that the SendInput function accepts. A union for the inputs that will be given and an integer indicating the type of input are both included in the INPUT struct. Check out this wiki if you want to learn more about unions. Let's start by putting the input structures KEYBDINPUT, MOUSEINPUT, and HARDWAREINPUT into practise.

Input Structs

KEYBDINPUT Struct:

Because the struct will be passed to unmanaged code, we will use Sequential StructLayout to force the members to be in sequential sequence. A virtual key code is wVk. The scan code for the key we wish to press is wScan. Here are further specifics about the scan codes. dwFlags are the input-related flags (KeyUp, ExtendKey, Unicode, ScanCode). To learn more

about the flags, kindly read the notes section below. Later, we'll get to watch them in action. Time is an input timestamp; if set to 0, the system generates its own timestamp. When using the GetMessageExtraInfo function, dwExtraInfo can be used to receive more details about a keystroke.

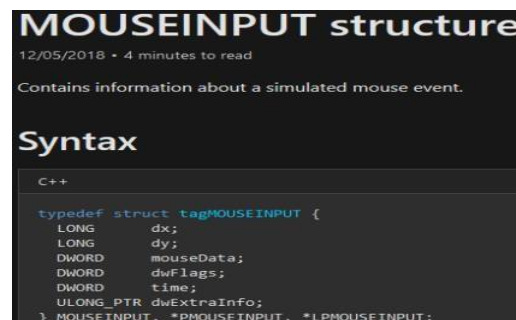


[StructLayout(LayoutKind.Sequential)]

```
public struct KeyboardInput
{
    public ushort wVk;
    public ushort wScan;
    public uint dwFlags;
    public uint time;
    public IntPtr dwExtraInfo;
}
```

MOUSEINPUT Struct:

Depending on the value of the dwFlags component, dx represents either the mouse's absolute position or the amount of movement since the last mouse event. The mouse's x position is used to specify absolute data, while the amount of pixels moved is used to specify relative data. For the y-axis, dy is the same as dx.



Mouse Data gives the amount of wheel movement if dwFlags contains MOUSEEVENTF WHEEL or MOUSEEVENTF HWHEEL. A positive value indicates that the wheel was rotated forward, away from the user; a negative value indicates that the wheel was rotated backward, toward the user. The definition of a wheel click is WHEEL DELTA, which is 120. A collection of bit flags called "dwFlags" describes different features of mouse motions and clicking buttons.

Later, we'll examine the flags. Time is an input timestamp; if set to 0, the system generates its own timestamp. An additional value connected to the mouse event is called dwExtraInfo. The GetMessageExtraInfo method is used to get it.

[StructLayout(LayoutKind.Sequential)]

```
public struct MouseInput
{
    public int dx;
    public int dy;
    public uint mouseData;
    public uint dwFlags;
    public uint time;
    public IntPtr dwExtraInfo;
}
```

HARDWARE INPUT Struct:

The message produced by the input hardware is known as uMsg. The low-order word of the lParam parameter for uMsg is called wParamL. The high-order word of the lParam parameter for uMsg is called wParamH.

[StructLayout(LayoutKind.Sequential)]

```
public struct HardwareInput
{
    public uint uMsg;
    public ushort wParamL;
    public ushort wParamH;
}
```


InputUnion:

The INPUT struct's union parameter is called InputUnion. The input data for the mouse, keyboard, or other hardware is contained in it.

```
[StructLayout(LayoutKind.Explicit)]
public struct InputUnion
{
    [FieldOffset(0)] public MouseInput mi;
    [FieldOffset(0)] public KeyboardInput ki;
    [FieldOffset(0)] public HardwareInput hi;
}
```

The INPUT Struct:

Contains data that is used by SendInput to synthesise input events such as keystrokes, mouse movements, and mouse clicks. The type of the input event is type. It details which input struct from the union will be used. the following values:

1. for MOUSEINPUT
2. for KEYBDINPUT
3. for HARDWAREINPUT

```
public struct Input
{
    public int type;
    public InputUnion u;
}
```

Flags:

A set of named constants of the underlying integral numeric type create an enumeration type, sometimes known as a "enum type," which is a value type (int, uint, byte, etc.). The associated constant values for enum members are by default of the type int, starting at zero and increasing by one in the sequence of the definition text.

Define enum members for those options so that each option is a bit field if you want an enumeration type to represent a combination of options. In other words, the values for those

enum items' related values should be powers of two. Then, you can combine options or intersect options by using the bitwise logical operators | or &, respectively.

[Flags]

```
public enum InputType
```

```
{
```

```
    Mouse = 0,
```

```
    Keyboard = 1,
```

```
    Hardware = 2
```

```
}
```



SHARON D

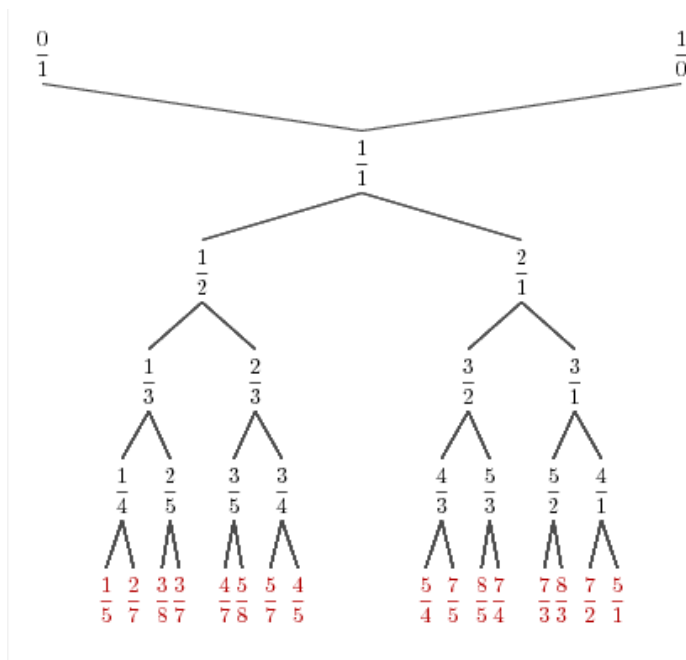
III-IT

STERN-BROCOT TREES AND THEIR APPLICATIONS

When attempting to prevent rounding errors in floating point, rational approximations are helpful. This article explains how to identify the "best" approximation.

Introduction:

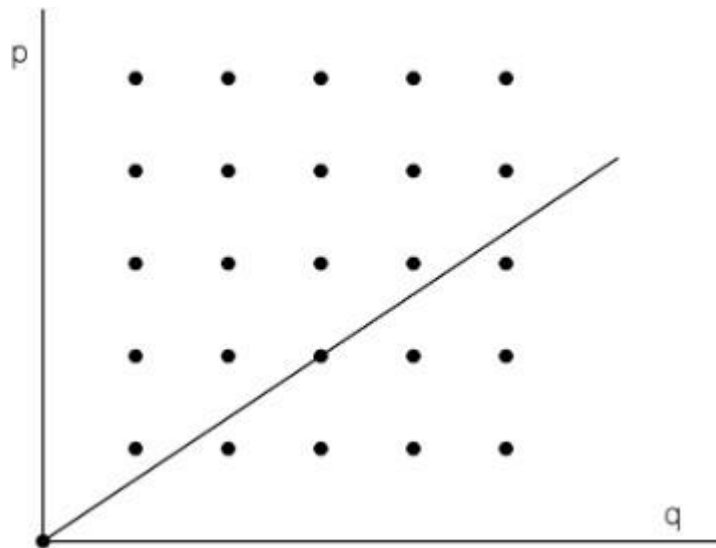
Sometimes, you can find things in math that are absolutely unrelated to what you were looking for. Achilles Brocot, a 19th-century French clockmaker, experienced this when he discovered the greatest rational approximation for irrational numbers while searching for the best approach to create geared wheels for his clocks. Why are you interested in this? Even though most modern processors include floating point units, working with integer numbers occasionally provides benefits. It's helpful to know that in certain circumstances, you can approximate as $355/113$, which is correct to seven decimal points. Stern and Brocot's tree structure can be used to find the answers. The illustration below shows the top several layers of this tree:



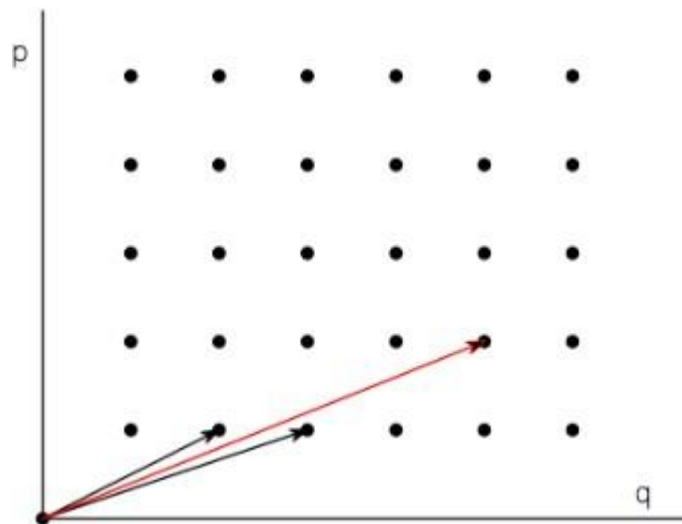
Background in Mathematics:

You can skip this section and move straight to the applications section if all you're interested in is the programming specifics. Any number that can be stated as a fraction (p/q) is considered rational. We shall only select the fraction in which P and q have no common factors because there are several fractions that can represent the same rational integer. For instance, the numbers $2/3$ and $4/6$ are equivalent but we will only use the first one. We can visualise rational

numbers as the slope of the line from the origin to the p/q point if we arrange the integer numbers on two axes. The representation of " $2/3$ " can be seen in the figure below.

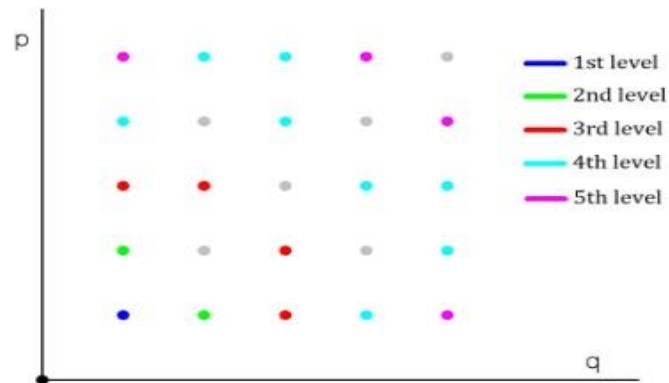


We need to define now the concept of median: the median of two rational numbers m/n and m'/n' is the rational number $(m+m')/(n+n')$. The numbers $1/2$ and $1/3$, together with their median, are shown below.



The median of those two numbers sits in between them when those numbers are viewed as a vector addition. In math terms, if $m/n < m'/n'$ then $m/n < (m+m')/(n+n') < m'/n'$;

Now that construction of the Stern-Brocot tree is complete. We begin with the digits 0 and 1, respectively. Although $1/0$ is not a true fraction, using it to represent "infinity" is nonetheless helpful. We add the medians of the numbers from the level before to each subsequent level of the tree. The median of those integers $(0+1)/(1+0) = 1/1$ is placed on level 2. We put the medians of each of those numbers on the following level: $(0+1)/(1+1) = 1/2$ and $(1+1)/(1+0) = 2/1$. The following figure demonstrates how the tree's subsequent layers spread out to encompass the grid of rational numbers.



Points that have common factors for p and q are those that are greyed out. We've seen how lines passing through one of the points of the integer grid can be used to represent rational numbers. Then, irrational numbers can be viewed as lines that almost always miss a grid point. Finding an approximate means locating a grid point that is somewhat close.

Stern-Brocot trees are used for rational approximations:

We've seen that our tree has the medians of the values from the level before on every level, and that a mediant's value lies between the values of its ancestors. The following approach can be used to search through such a tree for an adequate estimate of the number N with tolerance:

- ✚ Let T , the current node, serve as the tree's root.
- ✚ If the response is T and $|N-T|$ leave.
- ✚ Replace T with the left child of T if $N < T$, and with the right child of T in all other cases.
- ✚ Move on to step 2 now.

The features of the Stern-Brocot tree, or the algorithm, ensure that the approximation p/q discovered is the "best" in the sense that any other approximation p'/q' with the same precision will have $p' > p$ and $q' > q$.

Usage:

The best rational approximation with the specified precision is determined using the sba software. As for the command line:

```
sba <number> <precision>
```

```
~sba 3.14159265359 6
Finding approximation of 3.1415926536 with 6 decimals

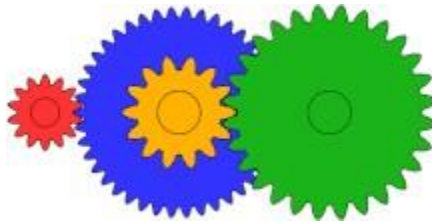
Current approximation 1/1 = 1.0000000 (err=2.14e+00)
Current approximation 2/1 = 2.0000000 (err=1.14e+00)
Current approximation 3/1 = 3.0000000 (err=1.42e-01)
Current approximation 4/1 = 4.0000000 (err=-8.58e-01)
Current approximation 7/2 = 3.5000000 (err=-3.58e-01)
Current approximation 10/3 = 3.3333333 (err=-1.92e-01)
Current approximation 13/4 = 3.2500000 (err=-1.08e-01)
Current approximation 16/5 = 3.2000000 (err=-5.84e-02)
Current approximation 19/6 = 3.1666667 (err=-2.51e-02)
Current approximation 22/7 = 3.1428571 (err=-1.26e-03)
Current approximation 25/8 = 3.1250000 (err=1.66e-02)
Current approximation 47/15 = 3.1333333 (err=8.26e-03)
Current approximation 69/22 = 3.1363636 (err=5.23e-03)
Current approximation 91/29 = 3.1379310 (err=3.66e-03)
Current approximation 113/36 = 3.1388889 (err=2.70e-03)
Current approximation 135/43 = 3.1395349 (err=2.06e-03)
Current approximation 157/50 = 3.1400000 (err=1.59e-03)
Current approximation 179/57 = 3.1403509 (err=1.24e-03)
Current approximation 201/64 = 3.1406250 (err=9.68e-04)
Current approximation 223/71 = 3.1408451 (err=7.48e-04)
Current approximation 245/78 = 3.1410256 (err=5.67e-04)
Current approximation 267/85 = 3.1411765 (err=4.16e-04)
Current approximation 289/92 = 3.1413043 (err=2.88e-04)
Current approximation 311/99 = 3.1414141 (err=1.79e-04)
Current approximation 333/106 = 3.1415094 (err=8.32e-05)

Found fraction 355/113 = 3.141593
Error= -2.67e-07
```


Epilogue - The Gears of Time:



You might be wondering why a clockmaker was interested in these trees after reading this tale. When two gears with p and q teeth are coupled together, their rotational ratio, or gear ratio, equals p/q . A double gear can be created by combining gears as in the example below:



In this instance, if the first couple's gear ratio (red/blue) is p_1/q_1 and the second couple's ratio (yellow/green) is p_2/q_2 , the total gear ratio for the first and second couples is $p_1/q_1 * p_2/q_2$.

Charles-Étienne-Louis Camus, a clockmaker from the 18th century, presented the following issue as an illustration of the difficulties clockmakers in the past faced: "To find the number of the teeth...of the wheels and pinions of a machine, which being moved by a pinion, placed on the hour wheel, shall cause a wheel to make a revolution in a mean year, supposed to consist of 365 days, 5 hours, 49 minutes.".

If we convert everything to minutes, the solution calls for creating a gear train with the ratio $720/525949$ (according to Camus, the average tropical year has 525949 minutes and the hour wheel completes one rotation in 12 hours, or 720 minutes).

BHUVANESH P

IV -IT

WHY SOFTWARE BUSINESSES FAIL

This article is a discussion of the reasons why businesses fail, with comparisons to ideas from Collapse: How Societies Choose to Fail or Succeed by Jared Diamond.

Disavowal:

My personal perspective regarding the causes of software firms' demise is expressed in this article. Since this is not a research paper, several of the statements have been made bold and directly to the point in order to keep the writing concise.

An Erupting Pan:

The software industry is a roiling pot of commerce. Some businesses, typically the larger ones, stall and eventually wither while others develop quickly, vanish almost instantly, and never come back. The software industry is not particularly stable. Companies that once dominated their respective markets (such as Netscape) no longer exist.

Societies and Businesses:

I needed to know whether a software corporation could be considered a small society and, if so, if the same principles applied. A society is, in the words of Wikipedia, "a group of people engaged in ongoing social interaction, or a sizable social group sharing the same physical or social region, usually subject to the same political authority and dominant cultural expectations. Societies are defined by patterns of relationships (social relations) between people who have a common culture and set of institutions; one society can be characterized by the whole of these relationships among the people who make up its membership.



A business, on the other hand, is defined as "a legal body reflecting a group of persons, whether natural or legal or a combination of both, with a definite aim. Members of the company

work together for a shared cause in order to accomplish clearly stated objectives. Even though they are worded differently, I see a lot of similarities between these two definitions. Societies and businesses both:

- ✓ are persons or groups of people;
- ✓ occupy a particular area of social territory;
- ✓ a managerial structure that is subordinate to authority;
- ✓ are bound by cultural norms (business procedures and behaviour regulations);
- ✓ bring people together to pursue a same objective (provide means to sustain existence of these individuals).



A Collection of Dark Tales:

In his interesting book *Collapse: How Societies Choose to Fail or Succeed*, Jared Diamond looked into a number of ancient civilizations that had successful expansion followed by a swift collapse. He developed a five-factor framework that incorporates the most important causes of social collapse by researching, among other civilizations, those of Easter Island, Chaco Canyon, the Maya, and Greenlandic Norse. For the reasons listed by Diamond:

- ✓ Society causes environmental harm, which shows up as environmental productivity that is below what is required to support the people (mainly deforestation, soil erosion, overhunting, overfishing and water mismanagement).
- ✓ The effects of climate change on food production.
- ✓ Increasingly antagonistic neighbors are endangering society physically.

- ✓ Collapse of supportive neighbors on whom society depends.
- ✓ Failure to alter activities in the face of the aforementioned threats.

Visualization of this idea is shown in Figure 1. The colour red denotes variables that society has no control over. The colour blue denotes socially determinable factors. By helping the neighbour in need, society can partially prevent the "collapse of friendly neighbours."



Figure 1. Jared Diamond's 5-factor framework.

Various Yet Similar:

I got to the conclusion that Diamond's theory can be modified to account for software company failures after pointing out commonalities between society and businesses. The revised framework is displayed in Figure 2.

Environment-related harm:

According to Jared Diamond, environmental degradation is the main cause of society's impending collapse. Soil erosion brought on by deforestation or, more generally, the destruction of natural vegetation was its main recurring manifestation. As a result, there were growing food shortages that ultimately led to famine, hunger, and civil conflict. This got me wondering what a software company's equivalent of soil would be. What provides nourishment for software businesses if dirt enables growing crops and puts food on tables? Features that can be sold to

consumers are the lifeblood of software companies, and new features "grow" within the existing codebase in a similar way that crops do in soil.

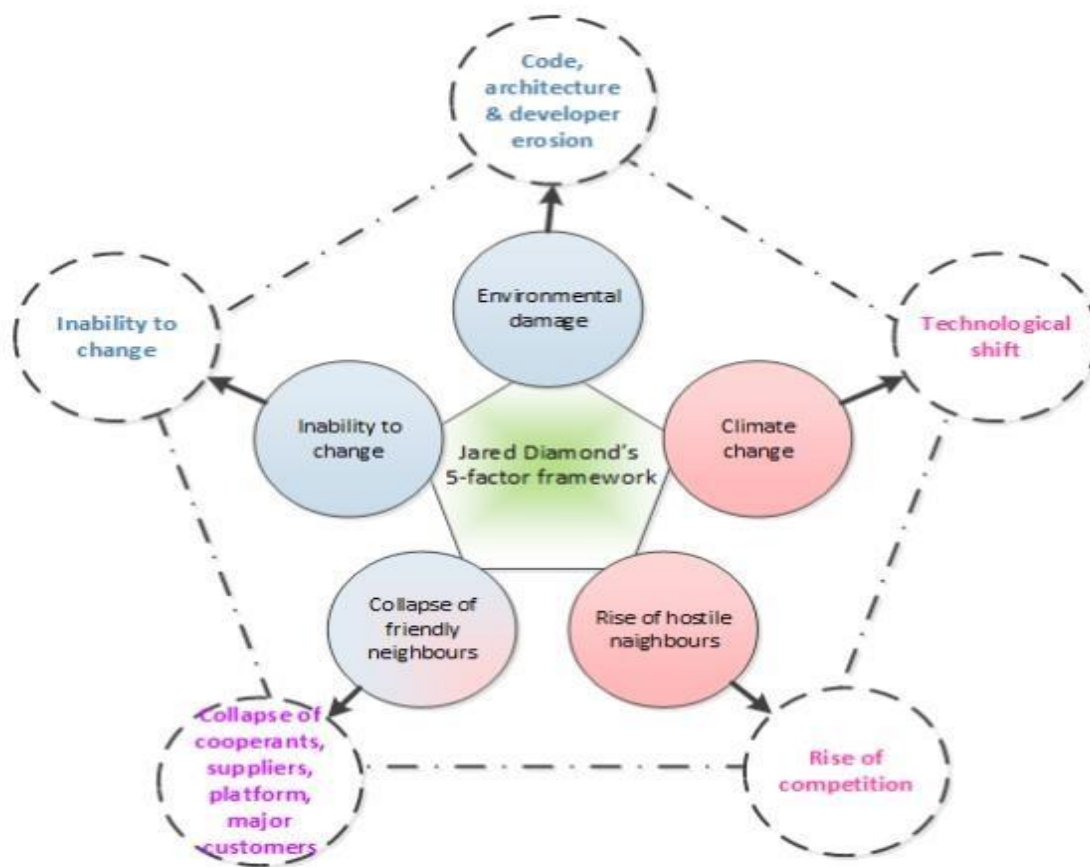


Figure 2. A 5-factor framework describing reasons for software company collapse.

This indicates that source code degradation, along with erosion of other artifacts like documentation and architectural integrity, is the equivalent of soil erosion. This comparison is not valid at the beginning of product development, before to initial release, but, in my opinion, it is valid for the majority of a software company's existence.

An eroded source code is one that is challenging to comprehend, hazy in function, disorganized in structure, and generally challenging to modify. A software corporation must continually tweak its old codebase in order to adjust its software product (to satisfy shifting market expectations), just like a society must grow crops in the same soil each year. When the soil is degrading, it requires more and more work to grow lesser crops in a year.

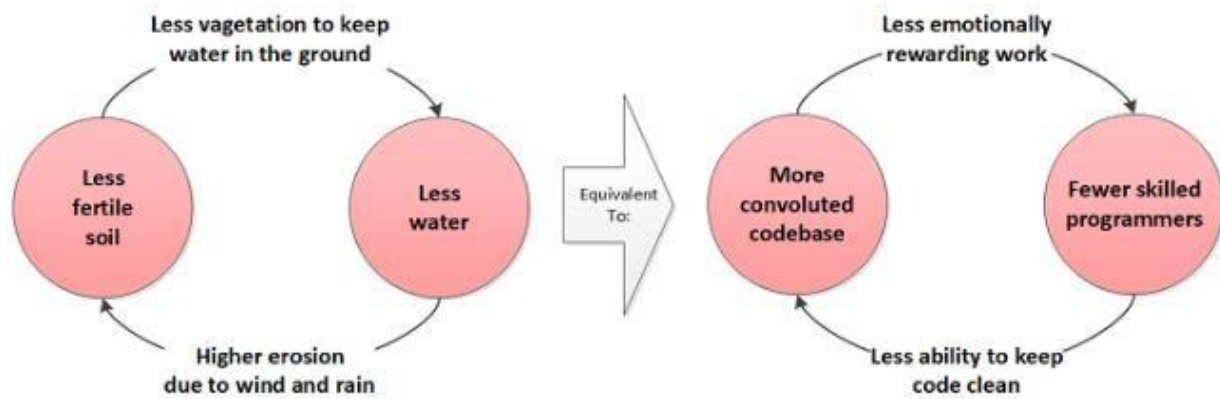


Figure 3. Vicious circle of erosion.

When a codebase is deteriorating, it takes an increasing amount of work to produce a decreasing number of modifications each year. If the process keeps going, costs will eventually exceed the value that was produced.

Water mismanagement is a problem that Diamond mentions as being closely related. Crops cannot be grown without good water, but what does water mean in the context of software development? The same way that new features and adaptations develop within an existing codebase thanks to programmers, much as crops grow in soil thanks to water (sunlight is accepted for granted).

SAKTHIVEL J

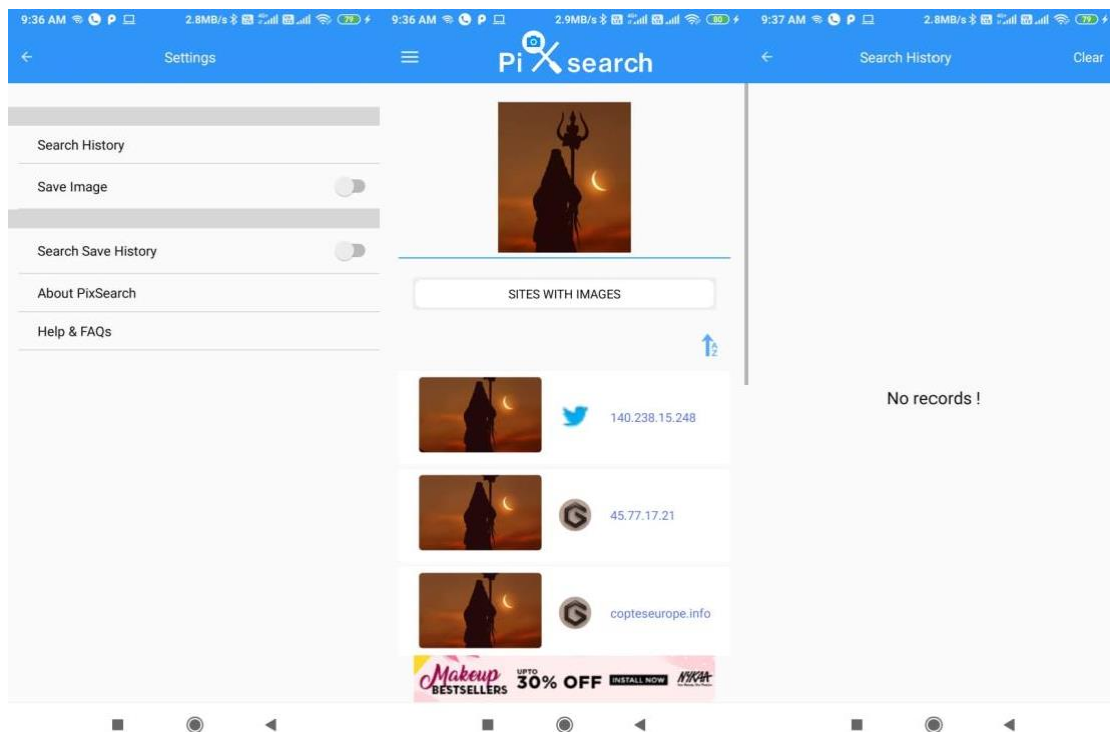
IV - IT

PROJECT GLIMPSES

PIX SEARCH

➤ It is a mobile search engine application that helps your web search using pictures instead of typing the text. Upload any pictures from your phone gallery or share image directly from websites to PixSearch app. The search displays all instances of the uploaded objects with a link to their websites. It's a great tool for comparing products from different websites before purchase. The search results will surprise you as it is very different to the traditional method. Search results are displayed for the image uploaded:

- All instances of the uploaded images are searched and displayed.
- Link to each of their website is also provided.
- Great tool to find “where used” of a given object or persons.
- For online shopping, great tool to find all sites selling the given product.



VIPASSANA IAS ACADEMY is committed to provide outstanding coaching and the best atmosphere to its aspirants through its commitment to providing value-based education and requirements, enabling them to fulfil the different needs and problems of society and nation.

- Commitment to providing high-quality education and improving administrative skills.
- Through academic efforts, continuous evaluation of teaching and learning processes is possible.
- Ensure that culture, ethics, and social responsibility are upheld throughout the academic process.

STUDENT PORTAL

Performance Home Dashboard Performance

Special Prelims Test Series 2021 Performance

Scores Attempted Progress Module Marks Progress Module Comparison with Toppers

| Choose Test Choose Subj Reset Filter | | | | | |
|--------------------------------------|-----------------|-----------|---------|-----------|-------|
| Subject Name | Total Questions | Attempted | Correct | Incorrect | Marks |
| General | 600 | 2 | 0 | 2 | 0 |
| CSAT | 240 | 0 | 0 | 0 | 0 |

Nature Wise Composition

General CSAT

Problem 1:

```
import java.util.Scanner;
public class Exercise1 {
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.print("Input an integer number: ");
        int n = in.nextInt();
        if (n <= 0) {
            System.out.print("Input a correct number.");
        }
        int x = 0;
        while (n != 1) {
            if (n % 5 == 0) {
                n /= 5;
            } else if (n % 3 == 0) {
                n /= 3;
            } else if (n % 2 == 0) {
                n /= 2;
            } else {
                System.out.print("It is not an ugly number.");
                x = 1;
                break;
            }
        }
        if (x==0)
            System.out.print("It is an ugly number.");
        System.out.print("\n");
    }
}
```

Sample Output:

Input an integer number: 235

It is not an ugly number.

Problem 2:

```
public class Example4 {
    public static void main(String[] args){
        int ctr = 0;
        int base = (args.length > 0) ? Integer.parseInt(args[0]) : 10;
        for(long n = 1; n <= 1000; n++){
            String sqr_Str = Long.toString(n * n * base);
```



```

        for(int j = 0; j < sqr_Str.length() / 2 + 1; j++){
            String[] parts = split_num(sqr_Str, j);
            long first_Num = Long.parseLong(parts[0], base);
            long sec_Num = Long.parseLong(parts[1], base);
            if(sec_Num == 0) break;
            if(first_Num + sec_Num == n){
                System.out.println(Long.toString(n, base) +
                    "\t" + sqr_Str + "\t " + parts[0] + " + " + parts[1]);
                ctr++;
                break;
            }
        }
    }
    System.out.println(ctr + " Kaprekar numbers.");
}

private static String[] split_num(String str, int idx){
    String[] ans1 = new String[2];
    ans1[0] = str.substring(0, idx);
    if(ans1[0].equals("")) ans1[0] = "0";
    ans1[1] = str.substring(idx);
    return ans1;
}
}

```

Sample Output:

```

1      1      0 + 1
9      81      8 + 1
45     2025     20 + 25
55     3025     30 + 25
99     9801     98 + 01
297    88209    88 + 209
703    494209   494 + 209
999    998001   998 + 001
8 Kaprekar numbers.

```

Problem 3:

```

import java.util.HashSet;
public class Example9 {
    public static void main(String[] args){
        System.out.println("First 10 Happy numbers:");
    }
}

```



```

    for(long num = 1,count = 0;count<8;num++){
        if(happy_num(num)){
            System.out.println(num);
            count++;
        }
    }
}

public static boolean happy_num(long num){
    long m = 0;
    int digit = 0;
    HashSet<Long> cycle = new HashSet<Long>();
    while(num != 1 && cycle.add(num)){
        m = 0;
        while(num > 0){
            digit = (int)(num % 10);
            m += digit*digit;
            num /= 10;
        }
        num = m;
    }
    return num == 1;
}
}

```

Sample Output:

First 10 Happy numbers:

1
 7
 10
 13
 19
 23
 28
 31

Editorial Faculty Member

Mr. B. Madusudhanan AP/IT

Editorial Students Member

Vimal S A (III – IT)

Srijena N (III – IT)

Abhirami B (II – IT)

Dinakaran M (II – IT)

Contact Us

Web: www.karpagamtech.ac.in

Address: L&T Bypass Road,
Bodipalayam post,
Coimbatore 641 105.