

Task 2: Using version control to collaborate on projects with multiple authors

In this task we will work with **github**. Even though there are many other version/revision control tools available, we'll be using **github**.

These are the reasons **github** is better suited to our needs:

1. It is free to use
2. Open source; available under GPL3
3. Centrally hosted. We need not host a git server ourselves
4. Is based on the **git** framework penned by **Linus Torvalds**, and thus fully supports **git**
5. In conclusion, github provides you all the benefits of git, without having to actually host a **git** server of your own. You get: ease of access, flexibility of use, instant availability, and unparalleled updation of patches

In this first task, we'll use the web interface of **github**, and we'll move onto the command line tool in the subsequent tasks.

To use github for project development, first you need to have a github account.

Creating an account is fairly simple and you can do it yourself, so I'll move on to the next step.

For the next step, follow this link: <https://github.com/SwapnilB31/MatrixTest>

This link will lead you to the Project Page of the repository named "MatrixTest".

Now we'll learn about the terminology.

1. **Repository:** Contains all the files related to your project. Usually comprises of just the source files and the readme document, but may also include things like stylesheets, icons, config files, screenshots, documentation etc. Anything and everything your project needs to work.
2. **Branch:** Branches represent different versions of the project. In a local setting (in your own repository), to make changes, you first create a new branch that has some name, and make the changes there. This ensures that, if

for some reason, the changes you made to your project cause it to become dysfunctional, or irreconcilable you always have the original source code intact and unchanged.

3. **Master Branch:** This is the root branch, and contains the accepted version of the project. When you create a repository, it is initialized with a master branch. All other branches are created and later merged with **master**.
4. **Pull Request :** Once you feel that you have successfully integrated the new features with the existing project, you can ask that these new features be pulled into the project. For this, you initiate a pull request, comparing your branch with the master. At this step, the interface will show you all the differences between this branch and the master, so that you can decide whether you want to push these changes or not. This is comparable to the **diff** command on **git-cli**. If you're sure, create a pull request.
5. **Merge Pull Request:** Once you have created a pull request, they will be visible under the pull request tab (if you own the repository, or have read access to the repository to which you issued a pull request). To merge this pull request to master, just click on the request header and you will be led to a page that will have a button that reads "**Merge Pull Request**". By clicking this button you can merge the changes into the application.
6. **Commit:** Every change you make to the repository is a commit. Whether it is creating a new file, editing an existing file, issuing pull requests, or creating branches, every modification is a commit. Commits have two description labels : a) Header – That provides a small headline describing the commit. b)Description(optional) – A detailed description of the changes that were made.

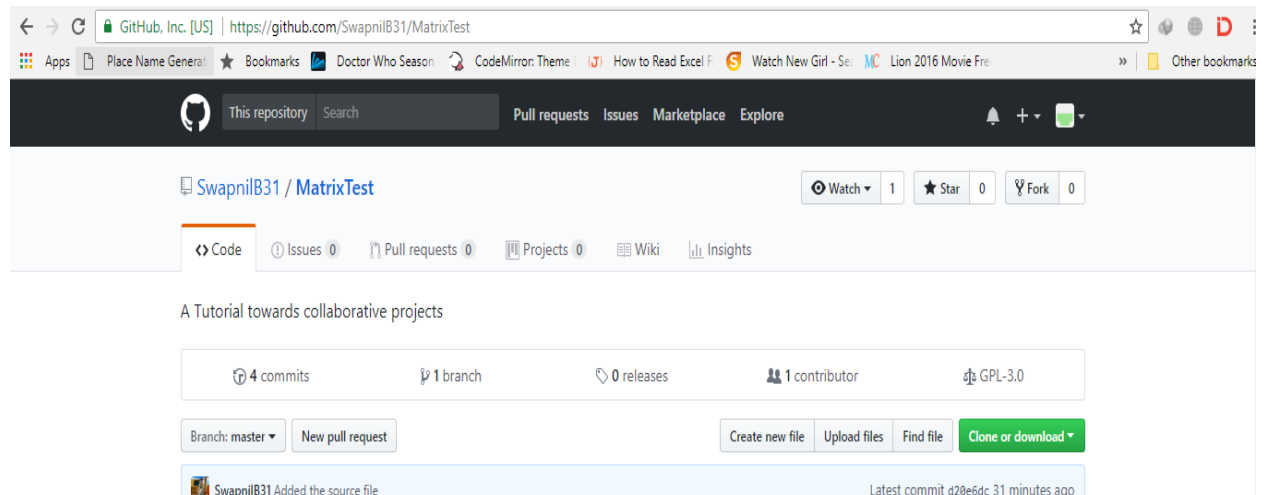
In an environment where many people are working with you it's important that you communicate with your co-contributors, when and why you made a change. SO make sure to at least briefly explain a commit in the header, even if you leave out the detailed description.

How Do I contribute to projects that I don't own or have write privileges to?

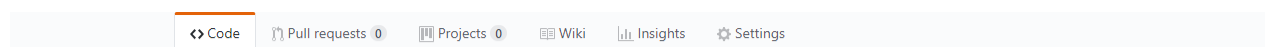
The answer to this question is the key to collaborating on projects. Most projects you want to collaborate on may not give you write access. In that case you have to follow these steps:

1. Fork the Project:

Click the fork button at the top of the Project Repository Page

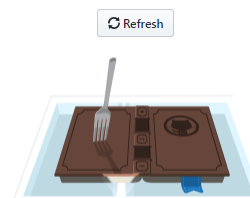


A copy of that project will added to your account. It will look something like this:



Forking SwapnilB31/MatrixTest

It should only take a few seconds.



Repository Forked, *Voila!*

The screenshot shows the GitHub repository page for 'InetZen / MatrixTest', which is a fork of 'SwapnilB31/MatrixTest'. The repository has 4 commits, 1 branch, 0 releases, 1 contributor, and is licensed under GPL-3.0. The 'Code' tab is selected, showing the 'master' branch. A table of commits is displayed, with the latest commit 'd20e6dc' made 35 minutes ago by 'SwapnilB31', adding the source file. The commit history includes 'LICENSE' (Initial commit, an hour ago), 'README.md' (corrected grammatical mistakes, 37 minutes ago), and 'main.cpp' (Added the source file, 35 minutes ago). The 'README.md' file is visible in the file list.

2. Create a branch to edit

Click on the branch tab to create a new branch (say **my-feature**)

The screenshot shows the same GitHub repository page, but with the 'Switch branches/tags' dialog open. The dialog has a search bar containing 'my-feature'. Below the search bar, there are tabs for 'Branches' and 'Tags'. A blue button labeled 'Create branch: my-feature from 'master'' is visible. The background repository page shows the same commit history, but the 'Branch: master' dropdown is now set to 'my-feature'.

And the branch will be created

The screenshot shows the GitHub repository page after the 'my-feature' branch has been created. The 'Branch: my-feature' dropdown is now selected, and the 'New pull request' button is visible. The commit history remains the same, with the latest commit 'd20e6dc' made 39 minutes ago. The 'main.cpp' file is still listed in the file list.

3. Write your code

You have two choices here :

1. Clone the repository (But that requires you to install **github for pc**)
2. Just copy the file you want to modify, to your PC. Run the program and test it. Make the changes (write your own code to supplement it).

Compile it and test whether your features have successfully integrated into the existing codebase. And if it has, just copy it back into the file on **git**.

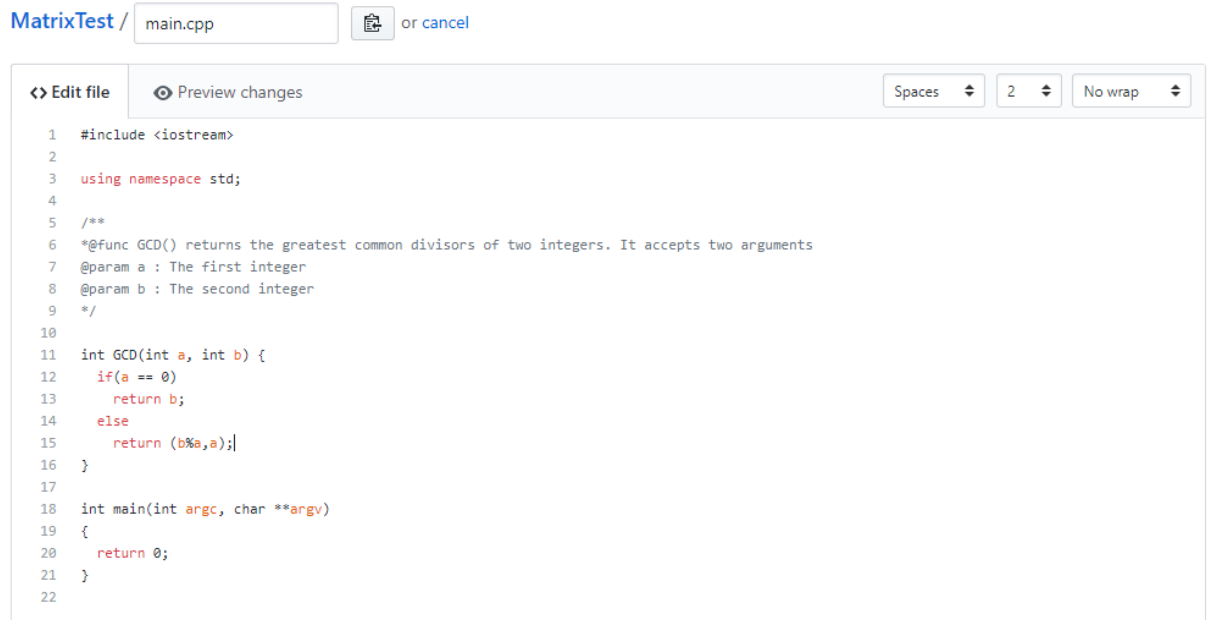
Clicking on the file displays its contents, and clicking on the pen icon, opens an editing tool. Just copy the existing code from this editor to your system, and afterwards, paste the modified code back into the editor.

To save the changes made you have to commit them.

Make sure that you are atleast adding a description header to your commit, so that others can understand you.

This is illustrated below:

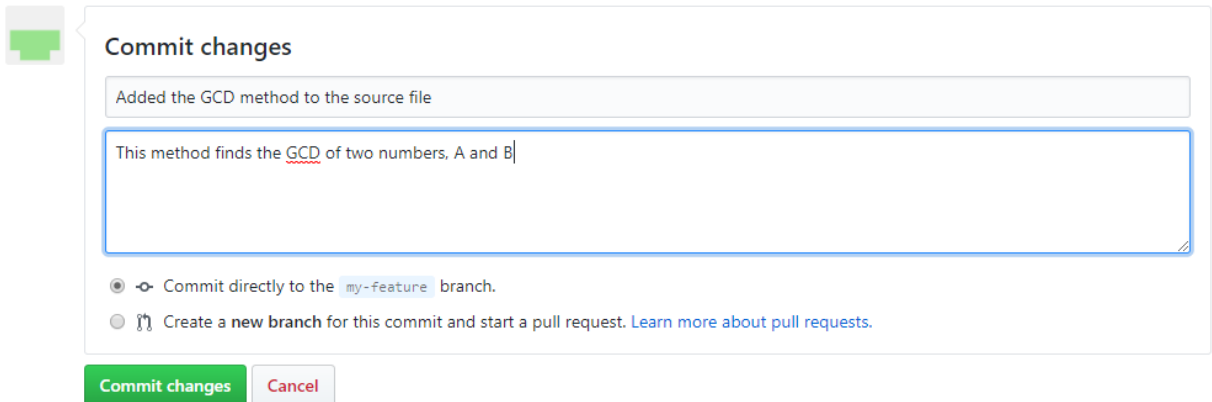
Adding the GCD function to the file main.cpp



The screenshot shows a code editor interface for a file named 'main.cpp'. The editor has a top bar with 'MatrixTest /' and a search bar containing 'main.cpp'. Below the top bar, there are tabs for 'Edit file' and 'Preview changes'. The code is displayed in a monospaced font with syntax highlighting. It includes a header for <iostream>, uses the std namespace, and defines a GCD function. The GCD function takes two integers, a and b, and returns the greatest common divisor. The main function calls GCD with arguments 10 and 15, and prints the result. The code is as follows:

```
1  #include <iostream>
2
3  using namespace std;
4
5  /**
6   * @func GCD() returns the greatest common divisors of two integers. It accepts two arguments
7   * @param a : The first integer
8   * @param b : The second integer
9   */
10
11 int GCD(int a, int b) {
12     if(a == 0)
13         return b;
14     else
15         return (b%a,a);
16 }
17
18 int main(int argc, char **argv)
19 {
20     return 0;
21 }
22
```

Committing the changes:



Commit changes

Added the GCD method to the source file

This method finds the GCD of two numbers, A and B

☒ Commit directly to the `my-feature` branch.

☐ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes **Cancel**

4. Create a pull request to 'master'

Create a pull request by clicking on the pull request tab. This is to merge the **my-feature** branch with master. Here you will compare master branch of your fork with the my-feature branch of your fork.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base: master ← compare: my-feature ✓ Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others.

↔ 1 commit 1 file changed 0 commit comments 1 contributor

Commits on May 16, 2018
InetZen Added the GCD method to the source file Verified eb6f82f

Showing 1 changed file with 13 additions and 0 deletions. Unified Split

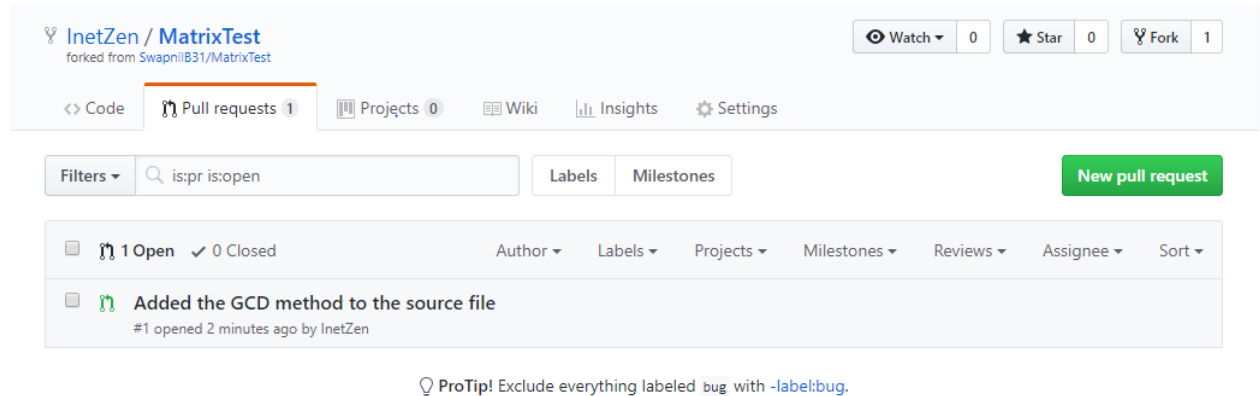
13 main.cpp View

2	2	@@ -2,6 +2,19 @@
3	3	using namespace std;
4	4	
5	5	/**
6	6	* @func GCD() returns the greatest common divisors of two integers. It accepts two arguments
7	7	* @param a : The first integer
8	8	* @param b : The second integer
9	9	*/
10	10	+
11	11	+int GCD(int a, int b) {
12	12	+ if(a == 0)
13	13	+ return b;
14	14	+ else
15	15	+ return (b%a,a);
16	16	+
17	17	+
18	18	int main(int argc, char **argv)
19	19	{
20	20	return 0;

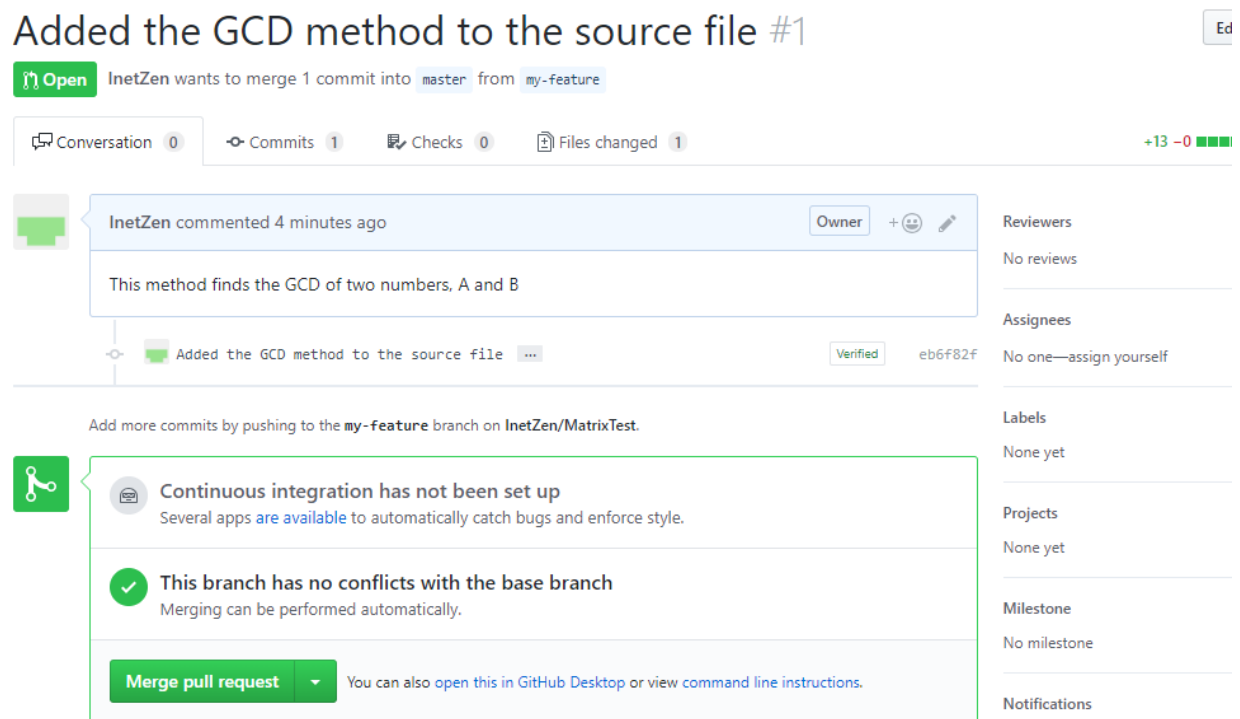
The changes that are new to the project are highlighted in green, as additions, and parts that are modified or removed are highlighted in red as deletions. This helps you verify whether these were the changes you wanted to make in the first place. If you're satisfied, then create a pull request.

5. Merge with master

Open the Pull Requests tab and you will see the request you created as open.



Click the header of the pull request and it will lead you to this page.



Click on the **Merge pull request** button to merge these branches

6. Create a pull request to the original repository

In order to make changes to the actual code base, you need to merge your changes to the source repository. The process is similar. Just the difference is that you compare your master branch with the source master branch.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base fork: SwapnilB31/MatrixTest base: master head fork: InetZen/MatrixTest compare: master

✓ Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others.

2 commits 1 file changed 0 commit comments 1 contributor

Commits on May 16, 2018

- InetZen Added the GCD method to the source file [Verified](#) eb6f82f
- InetZen Merge pull request #1 from InetZen/my-feature [Verified](#) c796818

Showing 1 changed file with 13 additions and 0 deletions. Unified Split

13 main.cpp View

Describe the changes made, and the pull request will be issued.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base fork: SwapnilB31/MatrixTest base: master head fork: InetZen/MatrixTest compare: master

✓ Able to merge. These branches can be automatically merged.

Addition of the GCD Feature

Write Preview

A new method that calculates the GCD of two numbers has been added to the source code.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

☒ Allow edits from maintainers. [Learn more](#) Create pull request

2 commits 1 file changed 0 commit comments 1 contributor

7. Wait for the Pull request to be accepted

Now you can wait for the owner of the repo to acknowledge this request. And when they do, your changes will be applied to the actual project.

On the project owner's end, they will receive a pull request. It will look like this:

SwapnilB31 / MatrixTest

Unwatch 1 Star 0 Fork 1

Code Issues 0 Pull requests 1 Projects 0 Wiki Insights Settings

Label issues and pull requests for new contributors
Now, GitHub will help potential first-time contributors discover issues labeled with **help wanted** or **good first issue**

Filters is:pr is:open Labels Milestones New pull request

1 Open 0 Closed Author Labels Projects Milestones Reviews Assignee Sort

Addition of the GCD Feature
#1 opened 16 seconds ago by InetZen

Conversation 0 Commits 2 Checks 0 Files changed 1

InetZen commented a minute ago First-time contributor

A new method that calculates the GCD of two numbers has been added to the source code.

InetZen added some commits 22 minutes ago


- Added the GCD method to the source file Verified eb6f82f
- Merge pull request #1 from InetZen/my-feature Verified c796810


Add more commits by pushing to the **master** branch on InetZen/MatrixTest.

Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

 SwapnilB31 merged commit `aba8a87` into `SwapnilB31:master` 13 seconds ago [Revert](#)




WritePreview

AA[▼] B *i* “ <> 🔗 ☰ ☷ ☰ ☷ ☷ ↶ @ 📌


Addition of GCD Method accepted

Attach files by dragging & dropping or [selecting them](#).

 Styling with Markdown is supported [Comment](#)

<> Code ⓘ Issues 0 🔗 Pull requests 1 📁 Projects 0 📖 Wiki 📊 Insights ⚙ Settings

Addition of the GCD Feature #1

 **Merged** SwapnilB31 merged 2 commits into `SwapnilB31:master` from `InetZen:master` a minute ago

Once your pull request has been accepted, you will receive an acknowledgement on your registered mail.

P.T.O.

Problem Statement:

Write a program in C++ that performs various operations on Matrices and augmented matrices provided by the user. This program has been broken down into several methods, one amongst which each member, without prior experience with github, has to implement and then push into the base repository at <https://github.com/SwapnilB31/MatrixTest>, by following the instructions provided above.

Note : 1. The dimension of the input matrix will be $m \times n$

2. Inverse can only be found if $m = n$

3. A system of linear equations can only be solved if :

a. It has m variables and;

b. It has m equations

Thus we require that $m = n$

4. For finding the solution to a system of linear equation

You can either use a matrix of dimension $m \times (m+1)$ or

A matrix of dimensions $m \times m$ and an array of size m

The following features are to be implemented. A member may choose one feature, implement it, and push it into the source program:

- 1) LCM of Two numbers
- 2) Cofactor of Matrix at index i,j
- 3) Minor of a Matrix at index i,j
- 4) Rotate a matrix clockwise by 90 degrees
- 5) Rotate a matrix anticlockwise by 90 degrees
- 6) Rotate a matrix clockwise by $90 \times n$ degrees, where n is a positive integer (May use method 4 to implement it, or otherwise write the code from the ground up)
- 7) Rotate a matrix anticlockwise by $90 \times n$ degrees, where n is a positive integer (May use method 5 to implement it, or otherwise write the code from the ground up)
- 8) Find the adjoint of a matrix
- 9) Find the determinant of a matrix
- 10) Find the inverse of a matrix using the determinant method

- 11) Transform a matrix into an identity matrix by elementary operations, and find its inverse (This method will receive two matrices : a. Input Matrix, b. Identity Matrix. You can use a technique called pivot and eliminate to achieve this)
- 12) Find the diagonal sum of a matrix
- 13) Find the upper triangle sum of a matrix
- 14) Find the lower triangle sum of a matrix
- 15) Transform a matrix into a scalar matrix by using elementary operations (Only values at the diagonal are non-zero)
- 16) Transform a matrix into an upper triangle matrix by using elementary operations
- 17) Solve a system of linear equations using the Gaussian elimination method
- 18) Solve a system of linear equations using the Gauss-Jordan elimination method
- 19) Solve a system of linear equations using Jacobi's method
- 20) Solve a system of linear equations using Gauss Seidel's method
- 21) Use a 2D array to create the forward difference table for given values of x and y
- 22) Create a Menu utility that will let the user pick a submenu to use and/or exit the program

Coding Conventions to be followed:

When a lot of people are contributing to a project, it is important that certain conventions be followed.

Since each person has his individual style, non-uniform naming conventions can create confusion and ambiguity.

The following naming conventions are to be followed:

- 1. All 2D Arrays are to be named 'mat'**
- 2. Solutions to system of linear equations must be named 'solArr'**
- 3. Indices of matrix should be referred to by integer variables 'i' and 'j'**

4. The number of rows will be denoted by an integer variable 'm'
5. The number of columns will be denoted by an integer variable 'n'
6. For all other identifiers use camel case.
7. Make sure that variable names, function names, class names etc. should be self-explanatory.

For instance, if you have an array on which you wish to take input, just name it 'inputArray'. Don't use vague names like 'a', 'arr', etc.

This reduces the need for documentation.

Thoroughly document your code. Writing comments explaining why you wrote crucial parts in the way you chose to write them.

Follow these conventions :

1. Before starting a function add a comment explaining what the function does, it's return value, what the arguments do, etc.
2. Use the annotation **@func** to describe a function
3. Use the annotation **@ret** to explain the return value
4. Use the annotation **@param** to describe the parameters/arguments to a method.

E.g.

```
/**
 *@func GCD() returns the greatest common divisors of two integers. It accepts two arguments
 @param a : The first integer
 @param b : The second integer
 @ret integer value : GCD of a and b
 */

int GCD(int a, int b) {
    /**
     *Uses the euclidean algorithm: Two assumptions :
     * 1. If we subtract smaller number from larger
     *    (we reduce larger number), GCD doesn't change.
     *    So if we keep subtracting repeatedly the larger of two,
     *    we end up with GCD.
     * 2. instead of subtraction, if we divide smaller number,
     *    the algorithm stops when we find remainder 0
     */

    if(a == 0)
        return b;
    else
        return (b%a,a);
}
```

Having thorough documentation helps when you need to debug code written by someone else. If the code is properly documented, then you need not spend time trying to understand the code. You can just jump right in and start debugging.

That's it for this task. Take your time and do it at your own peace. The goal of this task is to make you comfortable with writing code in a group environment. All the best.