Agenda
1. Input/Output Clarification
2. Simple Program (OOP)
3. std::vector

Oct 13
TA Session

Assignment 4: C++

ESS 201-Programming II

International Institute of Information Technology – Bangalore

Submission: Domjudge and LMS by 26 October 23:59:59

Problem:

In this assignment, you are going to model a Social Media Application, which consists of the following classes: System, Posts, and Users

Descriptions of the Classes:

- **System**

  Should maintain a list of all posts

  You may use std::vector for this

  Should maintain a list of all users

  You may use std::vector for this

  Should maintain the name of the application.

  You may use std::string for this

  (This should be initialised in a parameterised constructor.)

  Getters and Setters should be implemented wherever you see fit.

  Access Control:

  A User ID is passed into every method of this class (except for the Constructor). If the User ID provided does not belong to a User in the List of Users, the method should terminate at that point (Invalid User.)

- **User**

  Should maintain a unique User ID (statically generated by the User class.) - The values will be 1,2,3,...

  Should maintain a name

  You may use std::string for this

  Should maintain a list of users they follow (List of User objects)

  You may use std::vector for this

  Should maintain a list of their posts (List of Post objects)

  You may use std::vector for this

  Should have methods to create Posts and follow/unfollow another user

  Getters and Setters should be implemented wherever you see fit.

- **Post**

  Should maintain a unique Post ID (statically generated by the Post class.) - The values will be 1,2,3,...

  Each post will have a unique User who posts - the class will maintain a User ID Attribute.

  Should maintain content

  You may use std::string for this

  Should maintain a year of posting

  Getters and Setters should be implemented wherever you see fit.

  Initialise posts with the help of parameterised constructors

Input/Output Format:

All inputs are taken through std::cin and all outputs involve std::cout

Each line of the input will contain a keyword(in caps) followed by arguments. (Note: If an operation is invalid, print Invalid Input then and there and continue printing from the next line.)

CREATE nameOfSystem

USER usernameOfNewUserCreated

FOLLOW usernameOfTheOneFollowing usernameOfTheOneBeingFollowed

POST usernameOfTheOnePosting YearOfPost ContentOfPost(can contain spaces)

EXIT

EXIT specifies the end of the input. You must call all destructors here.

You must first print the name of the System (i.e the parameter in CREATE)

At the end of the input, you have to print all the usernames , their user IDs, and the usernames of the other users they follow (separated by a line)

After this, you have to print all the posts' usernames, post IDs, and years of posting - in ascending order of their IDs (separated by a line)

Sample Input:

CREATE sma

USER username1

USER username2

FOLLOW username1 username2

FOLLOW username1 username3 Invalid Input

POST username1 2021 OOP is cool

POST username3 2020 This should not work Invalid Input

FOLLOW username2 username1

POST username2 2019 READMEs are cool

```
USER username3
EXIT
```

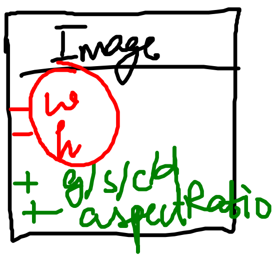
Sample Output:

```
sma
Invalid Input ✓
Invalid Input ✓
username1 1 username2
username2 2 username1
username3 3
username1 1 2021
username2 2 2019
```

.h
.c

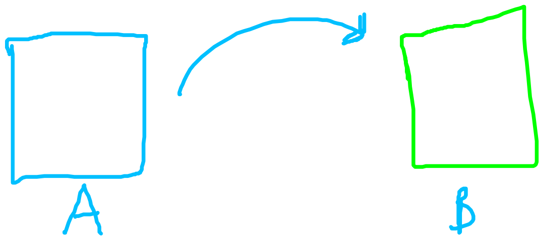
⊛ gscd

Simple OOP program



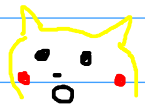
w/h

16:9
4:3



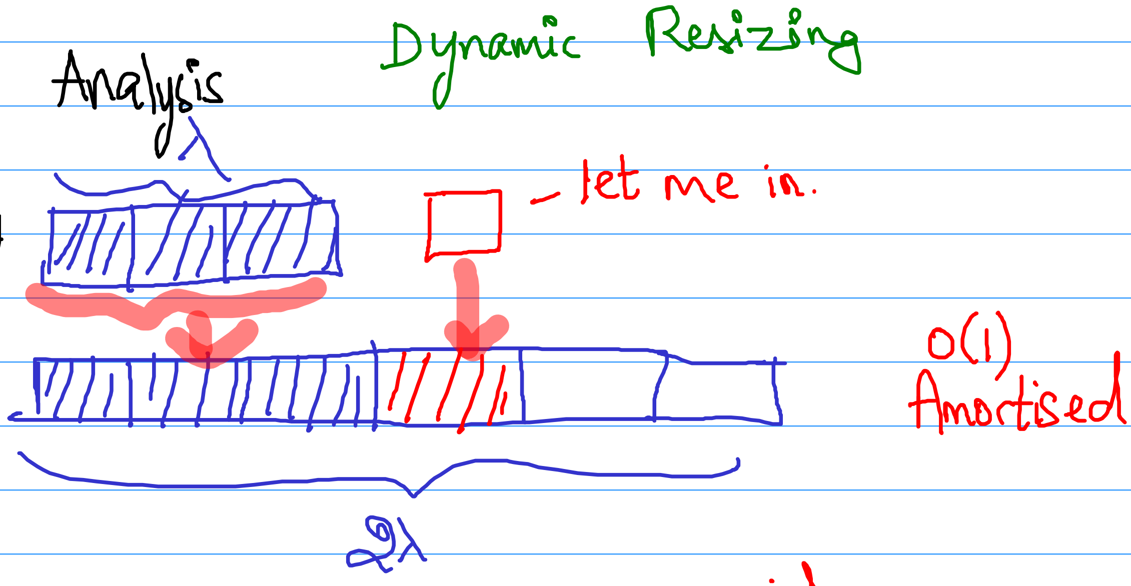
STL

- Template Functions
- Overpowered
- In lab4, we will be using std::vector (only vector is allowed)

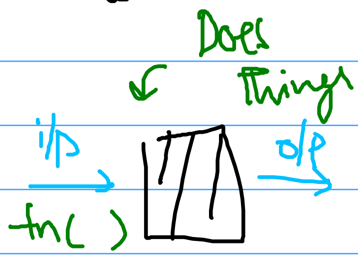


- Amortised Analysis

one of many awesome things in std::vector



- copied + moved



- Key Things
  - initialisation ✓
  - insertion ✓
  - deletion ✓
  - find elements ✓
  - iterators ✓

← pop(x)

What you need to know (for assignment at least)

- init( )
- push_back( )
- size( )
- erase( )
- at( )