`You'll submit the following file: Promotional.hpp Promotional.cpp Poll.hpp Poll.cpp General.hpp General.cpp Post.hpp Post.cpp`
`Please add documentation/comments almost in every line.`

Task 0: Usernames

The `Post` class must have the following additional *private member variables*:

A string representing the username of the Account that created it.
This will be useful in future projects.
The `Post` class must have the additional accessor and mutator *public member functions*:

```
/*
  Accessor Function
  @return      : username associated with this Post
*/
getUsername
/*
  Mutator Function
  @param       : a reference to the username associated with this Post
*/
setUsername
```

Task 1: Please subscribe and hit that Like button

Define and implement the `General` class as a subclass of `Post` (i.e. `General` inherits from `Post` it's public members).

Data and Types

The `General` class must have the following *type*

*** An Enum named Reactions with constants: LIKE, DISLIKE, LAUGH, WOW, SAD and ANGRY.**

The `General` class must have the following *private member variables*:

*** An integer array of size 6 that, indexed by Reactions, will store the number of reactions of each type the post has received.**

For example, the first element in the array will record how many reactions of type LIKE the post has received,

the second element will record the number of DISLIKE reactions the post has received, and so on.

The `General` class must have the following *public member functions*:

CONSTRUCTORS:

```
/**
  Parameterized constructor.
   @param    : The name of the General post
   @param    : The body of the General post

   @post     : Sets the title and body of the General post to the parameters.
               It will also generate the current time and store it
               and it initializes the array with default values
*/
```

IMPORTANT: **keep in mind the order in which constructors are called, which are superclass members and which are subclass members.**

UNIQUE METHODS

```
/**
  Mutator function to add a reaction
  @param    : A reference to reaction (represented by a value of type Reaction)
  @return   : return True if the react to the post is successful or false
              if the reaction provided is not within a valid Reaction value.

  @post     : Increments the array at the index corresponding to the input Reaction
*/
reactToPost

/**
  @post     : Prints the reaction to the post in this format (example):
              Like : 2 | Dislike : 0 | Laugh : 4 | Wow : 2 | Sad : 1 |  Angry : 5
```

***/**
getReactions

DISPLAYING
/**
@post : Displays the General post (example):
\n
{post_title_} at {time_stamp_}:
{post_body_}
\n
Like : 2 | Dislike : 0 | Laugh : 4 | Wow : 2 | Sad : 1 | Angry : 5
\n
***/**
displayPost

Task 2: And the results are in.....

Define and implement the `Poll` class as a subclass of `Post` (i.e. `Poll` inherits from `Post` it's public members).

Data and Types

The `Poll` class must have the following *private member variables :*
**** A vector of strings containing the poll options***
**** A vector of integers containing the number of votes for each poll option,***
where the integer in the first position indicates the number of votes for the poll option
in the first position , the second integer indicates the number of votes for the second
poll option, etc.
Note that, by this definition, the two vectors will have the same size!

The `Poll` class must contain public member functions that do the following:

CONSTRUCTORS:

/*
Parameterized constructor.

@param : The title of the Poll post
@param : The question of the Poll post
@param : A vector of options for the Poll post

@post :Sets the title and body of the Poll post to what was passed by the user.
It will also generate the current time and store it and will initialize the vectors of options and their respective number of votes.
/

IMPORTANT: ***keep in mind the order in which constructors are called, which are superclass members and which are subclass members***

UNIQUE METHODS:

/*

@param : a reference to int between 0 < number of options
@return : True if index is within range and we can vote for a poll, false otherwise
@post : Increments the poll votes based on the index which will refer to the index in the poll options
/
votePoll

/*
Mutator function used to either add a poll or change one of the poll options
@param : A reference to the new poll option
@param : A reference to int that can either represent the index of the existing option that will be replaced
or if choice_number > current number of options,
it will add this new option to the poll.

@post : Resets the number of votes for this option.
/
changePollOption

/*

```
    Accessor function
     @post      : prints the reaction to the post in this format
                (example where option_n is the string at position n in the poll options
vector):
                0 votes for: option_1
                5 votes for: option_2
                2 votes for: option_3
                ...
  */
  getPollOptions()


  /**
    Accessor function
    @param     : The index of the option
    @return    : returns the number of votes for a given option
  */
  getPollVotes
```

DISPLAY:

```
   /**
    @post   : displays the whole Poll post (example):
              \n
              {post_title_} at {time_stamp_}:
              {post_body_}
              \n
              0 votes for: option_1
              5 votes for: option_2
              2 votes for: option_3
              ...
              \n
  */
  displayPost
```

Task 3: Please disable Adblock before continuing

Define and implement the `Promotional` class as a subclass of `Post` (i.e. `Promotional` inherits from `Post` it's public members).

DATA AND TYPES
The `Promotional` class must have the following private member variables:
string that represents a url
The `Promotional` class must contain public member functions that do the following:

CONSTRUCTOR: ***/****
Parameterized constructor.
@param : The name of the Promotional post
@param : The body of the Promotional post
@param : The link of the Promotional post

@post : Sets the title, body and link (if it's a proper link otherwise "Broken Link")
of the Promotional post to what was passed by the user.
It will also generate the current time and store it.
*******/*
IMPORTANT: ***keep in mind the order in which constructors are called, which are superclass members and which are subclass members***

UNIQUE METHOD:
/*
Accessor function
@return : Returns the post link
*******/*
getLink
/*
@param : A reference to the link that is in the format 'https://www.something.something'
or 'http://www.something.something' with the last 'something' being at least 2 characters
@return : true if the link is correctly formatted, false otherwise

@post : If the link is not correctly formatted, store "Broken Link"

in the link member variable, otherwise store the value of the parameter (Hint: see <regex>)

**/*

setLink

DISPLAY:

*/***

@post : displays the entire Promotional post (example):

\n

{post_title_} at {time_stamp_}:

{post_body_}

\n

{link}

\n

**/*

displayPost