Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Fall 2022

Homework 5 – SUBook

Due: 13/12/2022, Tuesday, 11:55

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CAN NOT HELP any friend while coding.
Plagiarism will not be tolerated!!

Submission rules (PLEASE READ, IMPORTANT)

Your submission will consist of only TWO code files:

1. `SUBook.h`
2. `SUBook.cpp`

Please don't change the names of these files.

Don't upload an XCode project or a Visual Studio project. Just the two code files above.

Place these two code files AND ONLY THESE TWO FILES inside a folder named with the following convention:

`SUCourseUserName_YourLastname_YourName_HWnumber`

Your SUCourse user name is your SUNet username that is used for your sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the folder name. For example, if your SUCourse username is cago, your first name is Taha Çağlayan, and your last name is Özbugsızkodyazaroğlu, then the folder name must be:

`cago_Ozbugsizkodyazaroglu_TahaCaglayan_1`

Compress this folder using a compression program such as WinZip or WinRAR. Please use "zip" compression. "rar", "7z" or any other compression mechanisms are NOT allowed. Our homework processing system only works with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all your code files. You will receive no credits if your compressed folder does not expand or it does not contain the correct files. The name of the zip file follows the same convention. The zip file for the homework submission by the student mentioned above would be:

```
cago_Ozbugsizkodyazaroglu_TahaCaglayan_1
```

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.). Successful submission is one of the requirements of the homework. If for some reason you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Overview

For this assignment, you will be implementing a social media network that allows users to create posts and comment on these posts. The main classes you will be working with are User, Post, and Comment.

The User class should contain the following private data members:

- name: the user's name as a string.
- posts: a list of all the posts created by the user.
- num_comments: number of comments that the user made on all posts.

The Post class should contain the following private data members:

- user: the user who created the post.
- content: the text content of the post.
- comments: a list of all the comments on the post.

The Comment class should contain the following private data members:

- user: the user who made the comment.
- content: the text content of the comment.
- post: the post the comment is on.

The User, Post, and Comment classes should be able to share objects with each other. For example, a User object should be able to access a Post object's comments. Similarly, a Post object should be able to access a Comment object's user and content.

The data types of each data member are left for you to choose. As long as the classes you create work with the given main programs and there are no memory leaks, then you are good to go. Some data members are better to be stored as references, some as pointers (smart or regular), and some as values. The choice is yours.

You should also implement the following functions for each class:
<u>User:</u>

- A constructor that takes the user's name as a parameter.
- A destructor that will delete all the posts of this user.
- `Post& createPost(string content)`: creates a new `Post` object with the given content and adds it to the user's posts list. The function returns a reference to the created post.
- `vector<Post*> getPosts()`: return a vector of pointers to the posts of the user.

- `int getNumComments()`: returns the number of comments made by the user. If the user doesn't have any comments yet, the function returns 0.
- `string getName()`: get the name of the user.

Post:
- A constructor that takes as parameters the user that made the post and the text content of the post.
- A destructor that will delete all the comments on this post.
- `void addComment(string content, User* commenter)`: adds a comment by "commenter" with the content given.
- `vector<Comment*> getComments()`: get a vector of pointers at the comments on this post.
- `User& getUser()`: returns a reference to the user who made the post.
- `string getContent()`: return the text content of the post.

Comment:
- A constructor that takes as parameters the user that made the comment, the post on which the comment is on, and the text content of the comment.
- A destructor that decrements the number of comments of the user that made this comment by 1.
- `User& getUser()`: returns the user who made the comment
- `string getContent()`: returns the text content of the comment
- `Post& getPost()`: returns the post that this comment is on.

You will write two files. SUBook.h and SUBook.cpp. SUBook.h will contain the class declarations of User, Post, and Comment, and SUBook.cpp file will contain the implementations of these classes. You may add additional functions to these classes as well as additional free functions outside these classes. Declarations of free functions should be in the SUBook.h file, and definitions should be in the SUBook.cpp file.

Important notes and rules:
- The lists in the above classes can be arrays, vectors, linked lists, or whatever you want. However, the return types of the functions given cannot be changed. So, for example, even if you implement your list of posts inside the User class as an array, the function getPosts() should return a vector of pointers.
- You may use friend classes and friend functions.
- You may add additional functions to the given classes.
- You may add additional free functions.
- You may assume the following: Comment objects will only be destroyed inside the destructor of Post. Also, Post objects will only be destroyed inside the destructor of User.

  An example of this is shown in Sample main 2. When User2 is deleted (because it went out of scope), the User destructor deleted User2's post. And when User2's post was deleted, the Post destructor deleted the comments on that User2's post. That includes User1's comment. That is why we see that User1 has 0 comments after User2 was deleted.

Sample main programs

Here are some sample main programs that use these classes and their expected output:

Sample main 1

main.cpp

```cpp
#include <iostream>
#include "SUBook.h"
using namespace std;

void printPost(Post& post){

   cout <<  "|||||||||||||||||||||||||||||||||||||||" << endl;
   cout << post.getUser().getName() <<":" << endl;
   cout << post.getContent()<< endl;
   vector<Comment*> comments = post.getComments();
   for (int i =0; i < comments.size(); i++){
       cout << "  " << comments[i]->getUser().getName();
       cout << ": " << comments[i]->getContent() << endl;
   }
   cout <<  "|||||||||||||||||||||||||||||||||||||||" << endl <<
endl;
}

void printUserNumComments(User& user){
 cout << "User " << user.getName() << " has " <<
user.getNumComments() << " comments" << endl << endl;
}

int main() {
   User chell("Chell");
   User openai("OpenAI");
   User glados("GlaDOS");
   Post& post1= openai.createPost("Science. For humanity.");
   post1.addComment("It's the other way around.", &glados);
   post1.addComment("You monster.", &glados);
   Post& post2 = glados.createPost("Test subjects wanted. ");

   printPost(post2);
   printUserNumComments(glados);
```

```
    post2.addComment("...", &chell);
    post2.addComment("Hired.", &glados);

    printPost(post2);
    printUserNumComments(glados);

    post2.addComment("Can I be a test subject?", &openai);
    post2.addComment("Stay in your lane, Skynet.", &glados);

    printPost(post2);
    printUserNumComments(glados);

    post2.addComment("I'll bring cake.", &openai);
    post2.addComment("Welcome to the team.", &glados);

    printPost(post2);

    return 0;
}
```

Output:

```
|||||||||||||||||||||||||||||||||||||||
GlaDOS:
Test subjects wanted.
|||||||||||||||||||||||||||||||||||||||

User GlaDOS has 2 comments

|||||||||||||||||||||||||||||||||||||||
GlaDOS:
Test subjects wanted.
  Chell: ...
  GlaDOS: Hired.
|||||||||||||||||||||||||||||||||||||||

User GlaDOS has 3 comments

|||||||||||||||||||||||||||||||||||||||
GlaDOS:
Test subjects wanted.
  Chell: ...
  GlaDOS: Hired.
  OpenAI: Can I be a test subject?
  GlaDOS: Stay in your lane, Skynet.
|||||||||||||||||||||||||||||||||||||||

User GlaDOS has 4 comments

|||||||||||||||||||||||||||||||||||||||
GlaDOS:
Test subjects wanted.
  Chell: ...
  GlaDOS: Hired.
  OpenAI: Can I be a test subject?
  GlaDOS: Stay in your lane, Skynet.
  OpenAI: I'll bring cake.
  GlaDOS: Welcome to the team.
|||||||||||||||||||||||||||||||||||||||
```

Sample main 2

main.cpp

```
int main(){
   User u1("User1");
   {
       User u2("User2");
       Post& p1 = u2.createPost("User2 post");
       p1.addComment("User1 comment", &u1);
       p1.addComment("User2 comment", &u2);
       cout << u2.getName() << " has " << u2.getNumComments() << "
comments" << endl;
       cout << u1.getName() << " has " << u1.getNumComments() << "
comments" << endl;
   }
   cout << u1.getName() << " has " << u1.getNumComments() << "
comments" << endl;
   return 0;
}
```

Output

```
User2 has 1 comments
User1 has 1 comments
```

```
User1 has 0 comments
```

Good luck!

Homework prepared by ChatGPT