CMPE 242
Spring 2021
Programming Homework #3

This assignment is due by 23:59 on Wednesday, May 19, 2021

Partner Selection Submission Due 23:59 on Thursday, May 6, 2021

You are welcome to ask your HW related questions. Please use one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:
   - 2021S_CMPE242_HW3-RECITATION1: May 6, 06:00-07:00 PM, Zoom ID: 99646340535
   - 2021S_CMPE242_HW3-RECITATION2: May 11, 06:00-07:00 PM, Zoom ID: 94751646079

Note: Please make sure that you have read the HW document well before participating.

Partner Selection(Due by 23:59 May 6, 2021)

For this assignment, you may work with one partner. If you do so, the two of you will turn in only one assignment and, except in extraordinary situations, receive the same grade (and, if applicable, have the same number of late days applied). Working with a partner is optional; it is fine to complete the assignment by yourself.

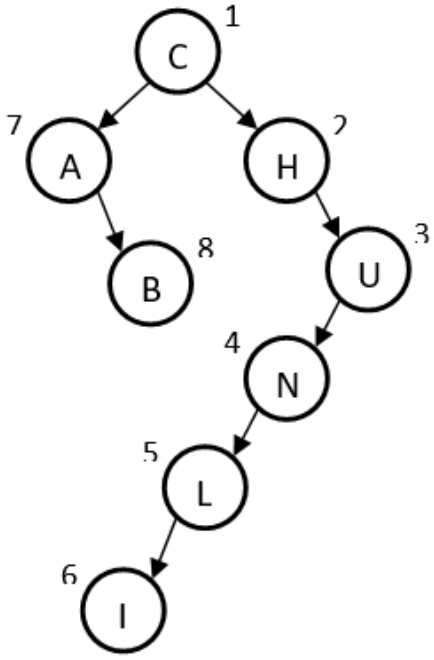
If you choose to work with a partner, **one of you must submit group names before the partner-selection due date**. If you choose to work alone, also submit to Moodle and inform us by submission. After the partner selection deadline, you are not allowed to make any change on how you will work on the assignment.

If you choose to work with a partner, you may divide the work however you wish, but both partners must understand and be responsible for everything that is submitted. The TA may meet with you to go over your code. Beyond working with a partner, all the usual collaboration policies apply.

PART I (20 point)

Build a binary tree for the **first 8 unique letters** of the concatenation of your name and surname. If your full name is shorter than 8 letters or it has not enough unique letters, append sufficient number of letters to the end of it, starting from the first letter of the alphabet (as an example, suppose that your name is CHUN LI, then build your tree with {C, H, U, N, L, I, A, B}; as another example suppose that your name is ALI CAN, then build your tree with {A, L, I, C, N, B, D, E}). Beginning with an empty binary search tree, insert the letters into the tree starting from the first letter. Indicate insertion order of each node with a number on the top of it. After drawing the tree, show the preorder, inorder, postorder, and level-order 1 traversals of your binary tree.

Example:



Pre-order: C – A – B – H – U – N – L – I

In-order: A – B – C – H – I – L – N – U

Post-order: B – A – I – L – N – U – H – C

Level-order: C – A – H – B – U – N – L – I

Part I of your homework should only contain a PDF file, which is 2 pages maximum. You may use handwriting and submit a scanned copy of your homework. No hardcopy submission or Java files are needed for this first part.

If you work with a partner, each of you should draw a separate tree related to your names, but you should submit a single PDF file; please also write the name of the partners on the PDF file.

PART II (80 point)

In this Part, you are supposed to implement a Java program for a movie database by using binary search trees.

In this movie database, for each movie, you will have an entry in which you keep:

1. The movie title,
2. The day, month, and year the movie was released to theaters,
3. The first name and last name of the director,
4. A list of its cast. For each actor/actress in the cast, you should keep:
    (a) The first and last name of an actor/actress, and
    (b) The title of the role s/he played in the movie.

In your implementation, you MUST keep the movie entries in a binary search tree by release year of the movie. Then, to keep the cast of each movie, you should create a binary search tree in which the actors are kept by their names (e.g., if you have five movies in your binary search tree, you will have five more binary search trees: one for the cast of each movie.). To deal with the same year movies, you may define your binary search tree such that for a node, smaller valued items are kept in the left subtree of the node and greater or equal valued items are kept in its right subtree.

Your system will have the following functionalities; the details of these functionalities are given below (see the end of this document for sample inputs/outputs):

1. Add a movie
2. Remove a movie
3. Add an actor/actress to the cast of a movie
4. Remove an actor/actress from the cast of a movie
5. Show the list of movies
6. Show detailed information about a particular movie
7. Query the roles that were played by a particular actor/actress
8. Query the movies which were directed by a particular director
9. Query the movies which are/were released in a particular year
10. Query the movies which are/were released between given years

Add a movie: This function adds an entry to the movie database for a given movie whose title, director's name (first name + last name), and release date (day + month + year) are specified as parameters. In this function, the cast is not specified; the actors/actresses that are in the movie will be added later. In this system, the titles of the movies are unique (i.e., no remakes of old movies are considered under the same title). Thus, if the user attempts to add a movie with an existing title, you should overwrite old information.

Remove a movie: This function removes a movie, whose title is specified as a parameter, from the movie database. If this movie does not exist in the database (i.e., if there is no movie with the specified title), you should not allow this operation and give a warning message.

Add an actor/actress: This function adds an actor/actress to the cast of a movie. For that, the title of the movie to which the actor/actress is added, the actor/actress' name (first name + last name), and the title of the role played by the actor/actress are specified as parameters. In this function, you should consider the following issues:

- If the movie with a specified title does not exist in the database, you should not allow the operation and give a warning message.
- In this system, all actor/actress' names are unique within the same cast. Thus, if the user attempts to add an actor/actress with an existing name **within the same cast**, you should overwrite old information. **Here note that an actor/actress can play in different movies.**

Remove an actor/actress: This function removes an actor/actress from the cast of a movie. For that, the title of the movie from which the actor/actress is removed and an actor/actress' name (first name + last name) are specified as parameters. If the movie with a specified title does not exist in the database, you should not allow the operation and give a warning message. Similarly, if the actor/actress is not in the cast of the specified movie, you should not allow the operation and give a warning message. Note that after calling this function, the cast should remain sorted in ascending order.

Show the list of movies: You should list all movie entries in the movie database on the screen in the following format. If the database does not contain any movie, you should display ---none---. Your system should display movies in **ascending** order by release year.

```
Title, release year, director name (for the 1st movie)
Title, release year, director name (for the 2nd movie)
Title, release year, director name (for the 3rd movie)
. . .
```

Show detailed information about a particular movie: You should display all of the information about a movie whose title is specified as a parameter. The output should be in the following format where the actor/actress' names should be sorted in alphabetically ascending order. If the movie with a specified title does not exist in the database, you should display ---none--- after the title. Your system should display actor/actress names in **ascending** order by actor name.

```
Title
Release day / release month / release year
Director name
Actor/actress name, title of his/her role (for the 1st actor/actress)
Actor/actress name, title of his/her role (for the 2nd actor/actress)
Actor/actress name, title of his/her role (for the 3rd actor/actress)
. . .
```

Query the roles that were played by a particular actor/actress: You should list all roles played by the actor/actress whose name (first name + last name) is specified as a parameter. The output should include the role, movie title, and the year, and should be in the following format. Note that if the actor/actress did not play a role in any movie, you should display ---none--- after the name of the actor/actress. Your system should display movies in **descending** order by release year.

```
Actor/actress name
Role, movie title, release year (for the 1st role)
Role, movie title, release year (for the 2nd role)
Role, movie title, release year (for the 3rd role)
. . .
```

Query the movies which were directed by a particular director: You should list all movies directed by the director whose name (first name + last name) is specified as a parameter. The output should include the titles and the release dates of the movies, and should be in the following format. Note that if the director did not direct any movie, you should display ---none--- after the name of the director. Your system should display movies in **descending** order by release year.

```
Director name
Movie title, release day / release month / release year (for the 1st movie)
Movie title, release day / release month / release year (for the 2nd movie)
Movie title, release day / release month / release year (for the 3rd movie)
. . .
```

Query the movies which are/were released in a particular year: You should list all movies released in the year which is specified as a parameter. The output should include the title of the movies, the director names, and release dates, and should be in the following format. Note that if there is no movie in the specified year, you should display ---none--- after the release year. Your system should display movies in **descending** order by release year.

```
Release year
Movie title, director name, release day / release month (for the 1st movie)
Movie title, director name, release day / release month (for the 2nd movie)
Movie title, director name, release day / release month (for the 3rd movie)
. . .
```

Query the movies which are/were released between particular years: You should list all movies released between (inclusively) the given years which are specified as two parameter. The output should include the title of the movies, the director names, and release years, and should be in the following format. Note that if there is no movie in the specified year, you should display ---none---. Your system should display movies in **descending** order by release year.

```
Released between year - year
Movie title, director name, release year (for the 1st movie)
Movie title, director name, release year (for the 2nd movie)
Movie title, director name, release year (for the 3rd movie)
. . .
```

The name of the class must be MovieDatabase, and must include these public member functions. We will use these functions to test your code. You can also define additional classes in your solution.

```
class MovieDatabase {
public:
       MovieDatabase();


       void addMovie( string movieTitle, string directorFirstName,
                      string directorLastName, int releaseDay,
                      int releaseMonth, int releaseYear );
       void removeMovie( string movieTitle );
       void addActor( string movieTitle, string actorFirstName,
                      string actorLastName, string actorRole );
       void removeActor( string movieTitle, string actorFirstName,
                         string actorLastName );
       void showAllMovies();
       void showMovie( string movieTitle );
       void showActorRoles( string actorFirstName, string actorLastName );
       void showDirectorMovies( string directorFirstName, string directorLastName );
       void showMovies( int releaseYear );
       void showMovies( int startYear, int endYear );
};
```

Example Test Code:

Below is an example of the test code that can be used for testing your programs. Of course, we will use other test codes in grading. Thus, do not forget to test your program with different test codes as well.

```
main()
{
     MovieDatabase md;

     md.showAllMovies();

     md.addMovie("Eyes Wide Shut", "Stanley", "Kubrick", 22, 10, 1999);
     md.addMovie("Family Plot", "Alfred", "Hitchock", 9, 4, 1972);
     md.addMovie("Psycho", "Alfred", "Hitchock", 20, 5, 1960);
     md.addMovie("Sweet and Lowdown", "Woody", "Allen", 26, 1, 1999);
     md.addMovie("Midnight in Paris", "Woody", "Allen", 30, 9, 2011);
     md.addMovie("Barton Fink", "Coen", "Brothers", 21, 8, 1991);
     md.addMovie("The Interpreter", "Sydney", "Pollack", 22, 4, 2005);
     md.addMovie("Psycho", "Alfred", "Hitchock", 20, 5, 1960);

     md.showAllMovies();
     md.removeMovie("Midnight in Paris ");
     md.showAllMovies();
     md.showMovie("Eyes Wide Shut");
     md.addActor("Barton Fink", "John", "Turturro", "Barton Fink");
     md.addActor("Barton Fink", "John", "Goodman", "Charlie Meadows");
     md.addActor("Barton Fink", "Judy", "Davis", "Audrey Taylor");
     md.addActor("Barton Fink", "Michael", "Lerner", "Jack Lipnick");
```

```
     md.addActor("Eyes Wide Shut", "Tom", "Cruise", " Bill Harford");
     md.addActor("Eyes Wide Shut", "Nicole", "Kidman", "Alice Harford");
     md.addActor("Eyes Wide Shut", "Madison", "Eginton", "Helena Harford");
     md.addActor("Eyes Wide Shut", "Jackie", "Sawaris", "Roz");
     md.addActor("Eyes Wide Shut", "Sydney", "Pollack", "Victor Ziegler");
     md.addActor("Midnight in Paris", "Woody", "Allen", "Woody Allen");
     md.addActor("The Interpreter", "Nicole", "Kidman", "Silvia Broom");
     md.addActor("The Interpreter ", "Sean", "Penn", "Tobin Keller");
     md.addActor("The Interpreter ", "Earl", "Cameron", "Zuwanie");
     md.showMovie("Barton Fink");
     md.showMovie("Eyes Wide Shut");
     md.removeActor("Eyes Wide Shut", "Jackie", "Sawaris", "Roz");
     md.showMovie("Eyes Wide Shut");
     md.showActorRoles("Nicole", "Kidman");
     md.showActorRoles("Judy", "Davis");
     md.showDirectorMovies("Alfred", "Hitchock");
     md.showDirectorMovies("Stanley", "Kubrick);
     md.showMovies(1999);
     md.showMovies(1972, 2005);

     exit();
}
```

Output of the Example Test Code:

For the test code that is given above, you will see the following output on the screen.

```
---none---

INFO: Movie Eyes Wide Shut has been added
INFO: Movie Family Plot has been added
INFO: Movie Psycho has been added
INFO: Movie Sweet and Lowdown has been added
INFO: Movie Midnight in Paris has been added
INFO: Movie Barton Fink has been added
INFO: Movie The Interpreter has been added
INFO: Movie Psycho overwritten

Psycho, 1960, Alfred Hitchock
Family Plot, 1972, Alfred Hitchock
Barton Fink, 1991, Coen Brothers
Sweet and Lowdown, 1999, Woody Allen
Eyes Wide Shut, 1999, Stanley Kubrick
The Interpreter, 2005, Sydney Pollack
Midnight in Paris, 2011, Woody Allen

INFO: Movie Midnight in Paris has been removed

Psycho, 1960, Alfred Hitchock
Family Plot, 1972, Alfred Hitchock
Barton Fink, 1991, Coen Brothers
Sweet and Lowdown, 1999, Woody Allen
Eyes Wide Shut, 1999, Stanley Kubrick
The Interpreter, 2005, Sydney Pollack
```

```
Eyes Wide Shut
22/10/1999
Stanley Kubrick

--none--

INFO: John Turturro has been added to the movie Barton Fink
INFO: John Goodman has been added to the movie Barton Fink
INFO: Judy Davis has been added to the movie Barton Fink
INFO: Michael Lerner has been added to the movie Barton Fink
INFO: Tom Cruise has been added to the movie Eyes Wide Shut
INFO: Nicole Kidman has been added to the movie Eyes Wide Shut
INFO: Madison Eginton has been added to the movie Eyes Wide Shut
INFO: Jackie Sawaris has been added to the movie Eyes Wide Shut
INFO: Sydney Pollack has been added to the movie Eyes Wide Shut
INFO: Movie Midnight in Paris does not exist
INFO: Nicole Kidman has been added to the movie The Interpreter
INFO: Sean Penn has been added to the movie The Interpreter
INFO: Earl Cameron has been added to the movie The Interpreter

Barton Fink
21/8/1991
Coen Brothers
John Goodman, Charlie Meadows
John Turturro, Barton Fink
Judy Davis, Audrey Taylor
Michael Lerner, Jack Lipnick

Eyes Wide Shut
22/10/1999
Stanley Kubrick
Jackie Sawaris, Roz
Madison Eginton, Helena Harford
Nicole Kidman, Alice Harford
Sydney Pollack, Victor Ziegler
Tom Cruise, Bill Harford

INFO: Jackie Sawaris has been removed from the movie Eyes Wide Shut

Eyes Wide Shut
22/10/1999
Stanley Kubrick
Madison Eginton, Helena Harford
Nicole Kidman, Alice Harford
Sydney Pollack, Victor Ziegler
Tom Cruise, Bill Harford

Nicole Kidman
Silvia Broom, The Interpreter, 2005
Alice Harford, Eyes Wide Shut, 1999

Judy Davis
Audrey Taylor, Barton Fink, 1991

Alfred Hitchock
Family Plot, 9/4/1972
Psycho, 20/5/1960

Stanley Kubrick
```

```
Eyes Wide Shut, 22/10/1999

1999
Sweet and Lowdown, 26/1
Eyes Wide Shut, 22/10

1972-2005
Family Plot, 1972
Barton Fink, 1991
Sweet and Lowdown, 1999
Eyes Wide Shut, 1999
The Interpreter, 2005
```

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59, on May 19, 2021.

2. You should submit your homework to course moodle page before the deadline. No hardcopy submission is needed. You should send files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

3. The below rules about late homework submissions apply. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. For this assignment, you are allowed to use the codes given in our textbook and/or our lecture slides. You ARE NOT ALLOWED to use any codes from somewhere else (e.g., from the internet, other text books, other slides ...).

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-------------------------------------------------
// Title: BST
// Author: Hamid AHMADLOUEI
// ID: 2100000000
// Section: 0
```

```
// Assignment: 1
// Description: This class defines a BST
//-------------------------------------------------------
```

9. Since your codes 1will be checked without your observation, you should report everything about your implementation. Well comment your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//-------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//-------------------------------------------------------
{
      // body of the function
}
```

- Indentation, indentation, indentation...

- This homework will be graded by your TAs, Hamid Ahmadlouei and İbrahim İleri (hamid.ahmadlouei@tedu.edu.tr, ibrahim.ileri@tedu.edu.tr). Thus, you may ask them your homework related questions. You are also welcome to ask your course instructor Tolga Çapın and Ulaş Güleç for help.

PART II GRADING RUBRICs (80 point)

Performance Element	Weight	Master (4/4)	Advanced (3/4)	Developing (2/4)	Beginner (1/4)	Insufficient (0/4)
Information	5%	Information (id, name, section, assignment number) given in each file.	Missing minor details.	Missing few details (e.g. section id).	Only name given.	None.
Documentation	5%	Every class and method has a detailed explanation (pre- and post-conditions,	Most classes and methods have an explanation, but missing in some parts.	Attempted to document the classes and methods, but they are not clear.	Only few comments.	Not even an attempt.

		sample I/O, etc).				
Code Design	5%	Modularity in source code and code format. Complete submission of classes and methods.	Methods make sense. Includes constructor that initializes carefully.	Uses set/get methods as necessary.	Class does very little; most functions remain in one main (driver) class.	Methods not properly defined.
Abstract Data Type: BST	10%	Uses BST structure for implementation own solution. And provides all functionality.	Uses tree data structure for own implementation. And provides most of expected functionalities.	Implements with major deviations from the specification.	Used different data structure to solve this problem.	Not even an attempt.
Functionality	50%	All functions are implemented with no missing functionality. Runs without any crash.	Missing some minor features or minor output problems. Runs without any crash.	Attempted to implement all functions but some of them do not work.	Only few functions are implemented correctly. Compiles but several warnings.	No working solution or does not compile.
Testing: Test data creation & generation	5%	Provided a tester class. Able to identify key test points. Clear on what aspects of the solution are being tested with each set. Able to generate test data automatically when necessary.	Provided a tester class. Able to identify key test points. Clear on what aspects of the solution are being tested with each set.	Provided a tester class. Able to identify key test points.	Sporadic.	No test case presented.