Dotify

Background Information

Spotify is one of the most popular music streaming services today, used by millions of users worldwide. However, before Spotify became the powerhouse that it is, it had a competitor that almost sunk it into the Silicon Graveyard—a product called Dotify. Dotify offered a service very similar to that of Spotify, but its main issue was that it couldn't understand the serious user need for spots instead of dots. Hence, its demise into the Silicon Graveyard. Your objective is to recreate the long forgotten Dotify product by maintaining a user's song library and playlists. (*Note: This may be a long specification, but over half of it is just examples about how the commands run. Don't feel overwhelmed. Read it carefully.)*

Basic Information

A song is uniquely identifiable by its name, the artist who performs it, and the album it belongs to. A song library is essentially a master list of all the songs owned by a user. The library must not contain duplicate songs. Each song in the library is assigned a positive integer valued identifier and has been played a certain number of times by the user. A playlist has a title, is assigned a rating from 1 to 5, and contains a subset of the library's songs. There cannot be playlists with duplicates titles.

Library File Format

A library data file contains all of the library's songs—one song on every line. Each individual component of a song's data is separated by a pipe character ("|") and is ordered as follows—name, artist, album, number of plays, and identifier. For example:

```
Dreaming of A Red Christmas|DJT|Take Over|2|18
Election Blues|The Dems|2016|107|39
War and Peace|The Confused|Super Confused|18|45
How?|The Confused|Very Very Confused|9|20
WHO DID IT|The Dems|2016|0|93
```

Playlist File Format

A playlist data file contains all of the playlists. The first line of a "playlist" contains a pipe character separated grouping of the title, the rating, and the number of songs in the playlist. Each subsequent line contains a pipe character separated grouping of a song's name, artist, and album. Note that the first line of each playlist has been bolded for clarity in the following example:

```
The Sadz|2|5
Dreaming of A Red Christmas|DJT|Take Over
Election Blues|The Dems|2016
War and Peace|The Confused|Super Confused
How?|The Confused|Very Very Confused
WHO DID IT|The Dems|2016
Happy Songs|5|1
Apples are a Way of Life|The Simons|The Fruit Album
Random|4|7
```

```
The Carrot|Sam Bundy|Vegetable Garden
Apples are a Way of Life|The Simons|The Fruit Album
Asymptote|The Mathematicians|Differentials
Holidays on Wheels|The Office|Scott's Tots
The Final Countdown|Simon Ayzman|Trash Talk
Apples in the Wind|Sam Bundy|Fruits of the Weather
Strawberry Typhoon|Sam Bundy|Fruits of the Weather
```

Program Startup

The Dotify program accepts two optional arguments of the form:

```
directory $ ./Dotify <library file name> <playlists file name>
```

These arguments function as initial "loading" steps that populate the user's library and playlists. The user can provide no arguments, one argument, or both arguments. If a provided file does not exist, skip over it. Assume that any test files provided by me do not contain duplicate songs or playlists, and that all expected values are valid (e.g., playlist rating is a number from 1 to 5). If (while reading in the playlists file) your program encounters a song in a playlist that does not exist in the library, skip that song and display the problematic line to the user. Some examples:

```
directory $ ./Dotify
No library file provided.
No playlists file provided.
(...normal program flow)

directory $ ./Dotify library.data
Loading library from "library.data".
No playlists file provided.
(...normal program flow)

directory $ ./Dotify library.data
Could not load library from "library.data". Skipping.
No playlists file provided.
(...normal program flow)

directory $ ./Dotify library.data playlists.data
Loading library from "library.data".
Could not load playlists from "playlists.data". Skipping.
(...normal program flow)

directory $ ./Dotify library.data playlists.data
Loading library from "library.data".
Loading playlists from "playlists.data".
Could not find song in library: "War and Peace|The Confused|Super Confused"
Could not find song in library: "How?|The Confused|Very Very Confused"
Could not find song in library: "WHO DID IT|The Dems|2016|0|93"
(...normal program flow)
```

Error Handling

For this project, you may **not** assume that any user input provided to your application's commands is valid input. If a user enters an invalid value, then you must abort the command with the message "`Invalid input.`" You must **not** abort the program.

Commands

Your application accepts the following set of commands:

COMMAND	DESCRIPTION

AS	This command adds a new song to the user's library. The user specifies the song's name, artist, and album. By default, the song is assigned the next available identifier and starts with 0 plays. If the song already exists in the library, indicate the problem.
	
	Examples
	`> AS`
	`What is the name of the song you'd like to purchase?`
	`> The Final Bloatdown`
	`Who is its artist?`
	`> Aimon Syzman`
	`Which album does it belong to?`
	`> Brash Balk`
	`"The Final Bloatdown" by Aimon Syzman (Brash Balk), identified as #13, already exists in your library.`
	
	`> AS`
	`What is the name of the song you'd like to purchase?`
	`> The Final Countdown`
	`Who is its artist?`
	`> Simon Ayzman`
	`Which album does it belong to?`
	`> Trash Talk`
	`"The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, purchased successfully to your library.`

RS	This command removes a song from the user's library and any playlists that that song belongs to. The user specifies the song's identifier. Display the playlists that the song was removed from. If the song does not exist in the library, indicate the problem.
	
	Examples
	`> RS`
	`What is the identifier of the song you'd like to remove from your library?`
	`> 28`
	`No song with identifier #28 exists in your library.`
	
	`> RS`
	`What is the identifier of the song you'd like to remove from your library?`

	`> 27`
	`"The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, removed successfully from your library.`
	
	`> RS`
	`What is the identifier of the song you'd like to remove from your library?`
	`of the song to remove?`
	`> 27`
	`"The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, removed successfully from your library and from playlists "Number Songs", "CS Jokes", "Trashy".`

PLY	This command plays a song in the user's library a certain number of times. The user specifies the song's identifier and the number of times to play the song. If the provided value is invalid, assume that the song was played 0 times. Display the number of plays started with and the new number of plays for that song. If the song does not exist in the library, indicate the problem.
	
	<u>Examples</u>
	`> PLY`
	`What is the identifier of the song you'd like to listen to?`
	`> 28`
	`No song with identifier #28 exists in your library.`
	
	`> PLY`
	`What is the identifier of the song you'd like to remove from your library?`
	`> 27`
	`How many times would you like to play this song?`
	`> blah`
	`"The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, played successfully 0 times (52 plays -> 52 plays).`
	
	`> PLY`
	`What is the identifier of the song you'd like to remove from your library?`
	`> 27`
	`How many times would you like to play this song?`
	`> -8`
	`"The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, played successfully 0 times (52 plays -> 52 plays).`
	
	`> PLY`
	`What is the identifier of the song you'd like to remove from your library?`
	`> 27`
	`How many times would you like to play this song?`
	`> 13`
	`"The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, played successfully 13 times (52 plays -> 65 plays).`

LB	This command displays all of the songs in the user's library. The user specifies the criterion to order the songs by—name, artist, album, or plays. The songs are displayed

	in a numbered list, with each displaying its name, artist, album, plays, and identifier.
	
	Examples
	`> LB`
	`You have no songs in your library.`
	
	`> LB`
	`What category should the songs be ordered by? (NAME/ARTIST/ALBUM/PLAYS)`
	`> NAME`
	`1. “Apples in the Wind” by Sam Bundy (Fruits of the Weather) - 6 plays [#11]`
	`2. “Asymptote” by The Mathematicians (Differentials) - 42 plays [#121]`
	`3. “Holidays on Wheels” by The Office (Scott’s Tots) - 35 plays [#57]`
	`4. “Strawberry Typhoon” by Sam Bundy (Fruits of the Weather) - 3 plays [#26]`
	`5. “The Carrot” by Sam Bundy (Vegetable Garden) - 151 plays [#9]`
	`6. “The Final Countdown” by Simon Ayzman (Trash Talk) - 10 plays [#666]`
	
	`> LB`
	`What category should the songs be ordered by? (NAME/ARTIST/ALBUM/PLAYS)`
	`> ARTIST`
	`1. “Strawberry Typhoon” by Sam Bundy (Fruits of the Weather) - 3 plays [#26]`
	`2. “Apples in the Wind” by Sam Bundy (Fruits of the Weather) - 6 plays [#11]`
	`3. “The Carrot” by Sam Bundy (Vegetable Garden) - 151 plays [#9]`
	`4. “The Final Countdown” by Simon Ayzman (Trash Talk) - 10 plays [#666]`
	`5. “Asymptote” by The Mathematicians (Differentials) - 42 plays [#121]`
	`6. “Holidays on Wheels” by The Office (Scott’s Tots) - 35 plays [#57]`
	
	`> LB`
	`What category should the songs be ordered by? (NAME/ARTIST/ALBUM/PLAYS)`
	`> ALBUM`
	`1. “Asymptote” by The Mathematicians (Differentials) - 42 plays [#121]`
	`2. “Strawberry Typhoon” by Sam Bundy (Fruits of the Weather) - 3 plays [#26]`
	`3. “Apples in the Wind” by Sam Bundy (Fruits of the Weather) - 6 plays [#11]`
	`4. “Holidays on Wheels” by The Office (Scott’s Tots) - 35 plays [#57]`
	`5. “The Final Countdown” by Simon Ayzman (Trash Talk) - 10 plays [#666]`
	`6. “The Carrot” by Sam Bundy (Vegetable Garden) - 151 plays [#9]`
	
	`> LB`
	`What category should the songs be ordered by? (NAME/ARTIST/ALBUM/PLAYS)`
	`> PLAYS`
	`1. “The Carrot” by Sam Bundy (Vegetable Garden) - 151 plays [#9]`
	`2. “Asymptote” by The Mathematicians (Differentials) - 42 plays [#121]`
	`3. “Holidays on Wheels” by The Office (Scott’s Tots) - 35 plays [#57]`
	`4. “The Final Countdown” by Simon Ayzman (Trash Talk) - 10 plays [#666]`
	`5. “Apples in the Wind” by Sam Bundy (Fruits of the Weather) - 6 plays [#11]`
	`6. “Strawberry Typhoon” by Sam Bundy (Fruits of the Weather) - 3 plays [#26]`

AP	This command creates a new playlist. The user specifies the playlist’s title. By default, the playlist starts off with a rating of 1 and no songs. The app must not contain duplicate playlists (i.e., playlists with the same title). If the playlist already exists, indicate the

	problem.
	
	Examples
	`> AP`
	`What is the title of the playlist you'd like to create?`
	`> The Sadz`
	`"The Sadz" playlist already exists.`
	
	`> AP`
	`What is the title of the playlist you'd like to create?`
	`> The Super Badz`
	`"The Super Badz" playlist created successfully.`

RP	This command removes a playlist. The user specifies the playlist's title. If the playlist does not exist, indicate the problem.
	
	Examples
	`> RP`
	`What is the title of the playlist you'd like to remove?`
	`> The Sadz`
	`"The Sadz" playlist does not exist.`
	
	`> RP`
	`What is the title of the playlist you'd like to remove?`
	`> The Super Badz`
	`"The Super Badz" playlist removed successfully.`

RN	This command renames an existing playlist. The user specifies the current title of the playlist to change and the new title. If the playlist does not exist or if a playlist with the specified title already exists, indicate the problem.
	
	Examples
	`> RN`
	`What is the title of the playlist you'd like to rename?`
	`> The Radz`
	`"The Radz" playlist does not exist.`
	
	`> RN`
	`What is the title of the playlist you'd like to rename?`
	`> The Sadz`
	`What is the new title that you'd like to rename it to?`
	`> The Madz`
	`"The Madz" playlist already exists.`
	
	`> RN`
	`What is the title of the playlist you'd like to rename?`
	`> The Sadz`
	`What is the new title that you'd like to rename it to?`
	`> The Super Sadz`

	"The Sadz" playlist renamed successfully to "The Super Sadz".
ASP	This command adds a song from the user's library to an existing playlist. The user specifies the playlist to add to and the song to be added. If the playlist does not exist, if the song does not exist in the library, or if the song being added already exists in the playlist, indicate the problem. > ASP What is the title of the playlist you'd like to add a song to? > The Super Madz "The Super Madz" playlist does not exist. > ASP What is the title of the playlist you'd like to add a song to? > The Super Radz What is identifier of the song to add to the playlist? > 28 No song with identifier #28 exists in your library. > ASP What is the title of the playlist you'd like to add a song to? > The Super Radz What is identifier of the song to add to the playlist? > 27 "The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, added successfully to playlist "The Super Radz".
RSP	This command removes a song from a playlist. The user specifies the playlist to remove the song from and the song to be removed. If the playlist does not exist or if the song does not exist in the playlist, indicate the problem. > RSP What is the title of the playlist you'd like to add a song to? > The Super Madz "The Super Madz" playlist does not exist. > RSP What is the title of the playlist you'd like to add a song to? > The Super Radz What is identifier of the song to add to the playlist? > 28 No song with identifier #28 exists in your library. > RSP What is the title of the playlist you'd like to add a song to? > The Super Radz What is identifier of the song to add to the playlist? > 27 "The Final Countdown" by Simon Ayzman (Trash Talk), identified as #27, removed successfully from playlist "The Super Radz".

RT	This command gives a new rating to a playlist. The user specifies the playlist to rate and a rating from 1 to 5. If the playlist does not exist, if the song does not exist in the playlist, or if the rating is invalid, indicate the problem.
	
	Examples
	`> RT`
	`What is the title of the playlist you'd like to rate?`
	`> The Madz`
	`"The Madz" playlist does not exist.`
	
	`> RT`
	`What is the title of the playlist you'd like to rate?`
	`> The Sadz`
	`What rating would you like to give this playlist? (1 to 5)`
	`> 0`
	`"0" is not a valid rating.`
	
	`> RT`
	`What is the title of the playlist you'd like to rate?`
	`> The Sadz`
	`What rating would you like to give this playlist? (1 to 5)`
	`> 4`
	`"The Sadz" playlist rated successfully as a 4.`

PL	This command displays all of the songs in a playlist. The user specifies the playlist. The songs are displayed in a numbered list in the order that they were added to the playlist, and each song shows its name, artist, album, number of plays, and identifier.
	
	Examples
	`> PL`
	`What is the title of the playlist you'd like to display?`
	`> The Madz`
	`"The Madz" playlist does not exist.`
	
	`> PL`
	`What is the title of the playlist you'd like to display?`
	`> The Radz`
	`"The Radz" playlist has no songs.`
	
	`> PL`
	`What is the title of the playlist you'd like to display?`
	`> The Sadz`
	`1. "Hillary O' Hillary" by Sam Bundy (2016) - 68 plays [#4]`
	`2. "Election Blues" by The Dems (2016) - 107 plays [#39]`
	`3. "Dreaming of A Red Christmas" by DJT (Take Over) - 2 plays [#18]`
	`4. "War and Peace" by The Confused (Super Confused) - 18 plays [#45]`
	`5. "How?" by The Confused (Very Very Confused) - 9 plays [#20]`

PLS	This command displays all of the playlists. The playlists are displayed in a numbered list ordered first by rating and then alphabetically by title. Each playlist shows its title, rating, and number of songs.
	
	Examples
	`> PLS`
	`1. "Happy Dance" – Rating: 5 – 25 songs`
	`2. "Unknown Artists" – Rating: 4 – 18 songs`
	`3. "The Radz" – Rating: 2 – 0 songs`
	`4. "The Sadz" – Rating: 2 – 8 songs`
	`5. "The Confused" – Rating: 1 – 4 songs`

AG	This command autogenerates a playlist. The user specifies the category to autogenerate by. By default, all autogenerated playlists start off with a rating of 1. If an autogenerated playlist has a title that already exists, indicate the problem.
	
	There are four possible autogeneration options: a song NAME, song ARTIST, song ALBUM, or MAGIC. If the user selects NAME, ARTIST, or ALBUM, then the user gives a specific query. All songs exactly matching that criteria get added to a new playlist titled with that specific query.
	
	For example, if the user selects NAME and provides the query **Love**, then all songs named **Love** will be added into a new playlist called **Love**. (It is not impossible for multiple songs in your library to have the same name, but different artists and albums.) If the user selects ARTIST and provides the query **Beyonce**, then all songs by the artist **Beyonce** will be added into a new playlist called **Beyonce**. If the user selects ALBUM and provides the query **Hamilton**, then all songs in the album **Hamilton** will be added into a new playlist called **Hamilton**.
	
	The fourth option is MAGIC. First, the algorithm detects (up to) the top 15 played songs in the user's library (if there are enough). If there are ties that bring the number over 15, then arbitrarily choose the songs such that the algorithm aggregates only 15 songs. Pick 5 random songs from your aggregated list. This is called the hit list.
	
	Starting with the first song in your hit list, pick a random component of the song (name, artist, or album). Find up to 3 other songs in your library (if they exist) that have the same value for that component. If more than 3 "similar" songs exist, arbitrarily choose any 3. These "similar" songs can include those already on the hit list. Finally, add these "similar" songs to a new playlist titled **AUTOGENERATED**. Repeat the aforementioned steps for the rest of the songs in the hit list, ultimately adding up to 3 "similar" songs to **AUTOGENERATED** for each hit list song.
	
	Examples
	`> AG`
	`What is the category you'd like to autogenerate a playlist with?`

```
(NAME/ARTIST/ALBUM/MAGIC)
> NAME
What is the song name you'd like to autogenerate a playlist from?
> Love
"Love" playlist already exists.

> AG
What is the category you'd like to autogenerate a playlist with?
(NAME/ARTIST/ALBUM/MAGIC)
> NAME
What is the song name you'd like to autogenerate a playlist from?
> Love
"Love" playlist autogenerated successfully.

> AG
What is the category you'd like to autogenerate a playlist with?
(NAME/ARTIST/ALBUM/MAGIC)
> ARTIST
Who is the song artist you'd like to autogenerate a playlist from?
> Beyonce
"Beyonce" playlist autogenerated successfully.

> AG
What is the category you'd like to autogenerate a playlist with?
(NAME/ARTIST/ALBUM/MAGIC)
> ALBUM
What is the song album you'd like to autogenerate a playlist from?
> Hamilton
"Hamilton" playlist autogenerated successfully.

> AG
What is the category you'd like to autogenerate a playlist with?
(NAME/ARTIST/ALBUM/MAGIC)
> MAGIC
Autogenerating based on the following songs:
    "Hillary O' Hillary" by Sam Bundy (2016). Similar songs by ALBUM:
        "Election Blues" by The Dems (2016).
        "Math Camp" by The Mathematicians (2016)
    "Apple on the Tree" by The Farmers (Fruit). Similar songs by NAME:
        (No similar songs founds)
    "The Carrot" by Sam Bundy (Vegetable Garden). Similar songs by ARTIST:
        "Apples in the Wind" by Sam Bundy (Fruits of the Weather)
    "Election" by The Confused (Super Confused). Similar songs by ALBUM:
        "How?" by The Confused (Super Confused)
    "Love" by The Beetles (Halp). Similar songs by NAME:
        "Love" by Simon Ayzman (CSCI 235 Blues)
        "Love" by Aimon Syzman (CSCI 235 Reds)
        "Love" by The Baetles (Emotional Attack)
"AUTOGENERATED" playlist autogenerated successfully.
```

EXP	This command exports the song library and playlists to files. The songs in the library will be exported to their own file and the playlists will be exported to their own file, based on the file format specified earlier. The user specifies the name of the library file and the name of the playlists file. If a file with a provided name already exists, then overwrite it. If the library contains no songs or if there are no playlists, then simply generate empty files.
	
	Example
	`> EXP`
	`What is the name of the file you'd like to export your library to?`
	`> library.data`
	`What is the name of the file you'd like to export your playlists to?`
	`> playlists.data`
	`Library and playlists exported successfully!`

HELP	This command displays all of the available commands and their descriptions.
	
	Example
	`> HELP`
	`AS: Purchases a new song to your library`
	`RS: Removes a specific song from your library`
	`AP: Creates an empty playlist`
	`RP: Removes a specific playlist`
	`RN: Renames a specific playlist`
	`AG: Autogenerates a playlist based on song name, artist, album, or magic`
	`ASP: Adds a specific song from your library to a playlist`
	`RSP: Removes a specific song from a playlist`
	`LB: Displays all the songs in your library`
	`PLS: Displays all the of the playlists in alphabetical order of title`
	`PL: Displays all the songs in a specific playlist in the order added`
	`RT: Rates a specific playlist from 1 to 5`
	`PLY: Plays a specific song in your library a specified number of time`
	`EXP: Exports the song library and playlists to files`
	`HELP: Displays this help menu`
	`EXIT: Exits the program`

EXIT	This command exits the program.
	
	Example
	`> EXIT`
	`Thank you for using Dotify!`

Program Completion Phases

First, you are responsible for looking through this specification and creating a design proposal outlining the classes you expect to have, the precise functionality that these classes provide, the data members/structures that these classes require, and how your classes interact with one another. Be sure to note whether any classes have inheritance relationships. Your design explanation should feature little to no code. It should instead focus on the macro-elements of your proposed program. You may

include UML diagrams. After I create your personal assignment repository, I will open up a GitHub Issue in the repository asking you to define the design of your project. You will give your design proposal there. See the Grading section to understand how this will be graded. After you submit your design proposal, I will give you feedback on your choices. Feel free to take my suggestions or not, and revise your design accordingly. You will then implement your program as per your (revised) design.

Provided Files

For this assignment, you are not given any starter code. You will be expected to implement all of the required code yourself. However, you will be given an executable file (a working solution based on this specification) and some test library/playlist files. Because the executable's solution code is compiled on the Linux lab machines, the executable can only be run on the Linux machines with confidence. Running it on any other system may yield undefined behavior.

Things to Think About

From an architectural perspective, there is no one perfect way to do this assignment. Nevertheless, good design choices can make your solution modular, extensible, efficient, and understandable. Endeavor to strike a balance that makes sense. Think about the following:

- What classes do you need to implement all of the required functionality? Is there inheritance? Which classes interact with the others, and how? Understand has-a vs. is-a relationships.
- Look at the STL documentation! Learn how to sort using the algorithms library. Learn about how to use iterators to loop through a data structure's elements. There's so a huge amount that the STL allows you to do. You just need to ask, explore, and research.
- Which data structures do you need to make use of? You have many data structure at your disposal now, so be sure choose them carefully. Justify your choices in the README.
- Which classes act merely as basic "data containers" and which classes are responsible for the logic of the Dotify's commands? Remember that a single class should handle a single logical set of responsibilities. Don't make any one class do too much heavy lifting. Be modular.
- Remember that your output format and style should match the given output format and style EXACTLY in order for you to receive full credit. Be sure to test your running program alongside the provided executable to make sure that yours conforms.
- Don't forget about memory management!

Submission Details

Your submission must include all of the header/source files (*.h/*.cpp) required for your program to properly compile and run. You must include a makefile that includes the sources to be compiled. The name of the generated executable must be **Dotify**. Feel free to adapt the makefile provided in Assignment #1. Do not submit any generated executables or object files (*.o). You must also update the README file with any guidance that someone looking at your project needs to and might like to know; this could include compilation instructions, interface specifications, interactions between classes, problems overcome, etc. Confirm that your code successfully compiles and runs on the Linux lab machines. See the **Assignment Guide Using Git & GitHub** for step-by-step information about how to submit your assignment via git and GitHub.

Due Dates

There are two due dates for this assignment. The due date for the program design proposal is **Tuesday, November 29th, by midnight**. I will not look at any design proposals submitted after this date. The earlier you submit your proposal, the earlier you will receive feedback from me. If you start coding before I give you feedback, your implementation may suffer as a result. The due date for the implementation portion of the assignment is **Saturday, December 17th, by midnight**. These are hard deadlines, meaning that there will be little leniency with regard to lateness. For every day that you miss the deadline for the implementation portion, I will deduct 10% from the project's final grade. A sample solution will be provided some time after the deadline.

Grading

Your grade is comprised of the following:

Components	Percentage	Relevant Questions
Correctness	50%	Do each of your commands provide output as expected based on the input?
Design	25%	Does your program design proposal separate out logical units into classes? Does it employ data structures that make sense and/or solve your problems efficiently?
Documentation	15%	Does your README document provide users with adequate information about your program? Do you adequately comment your class interface files and class implementation files (when necessary)?
Style	10%	Is your coding style readable and consistent? Do you follow (to the best of your ability) the modified Google Style Guide?

If your program does not compile on the Linux machines, you will receive no points for the **Correctness** component. Half of the **Design** component will be based on the robustness and clarity of your program design proposal. You will receive no credit for that half if you submit your proposal after the design proposal deadline.

Final Words

This assignment is the hardest one yet! Don't be discouraged. Discuss things over with your peers often. You may be tempted to plagiarize. Don't. I will catch you, just like I did the plagiarizers last semester. Start early. (You'll thank yourself later.) Good luck!