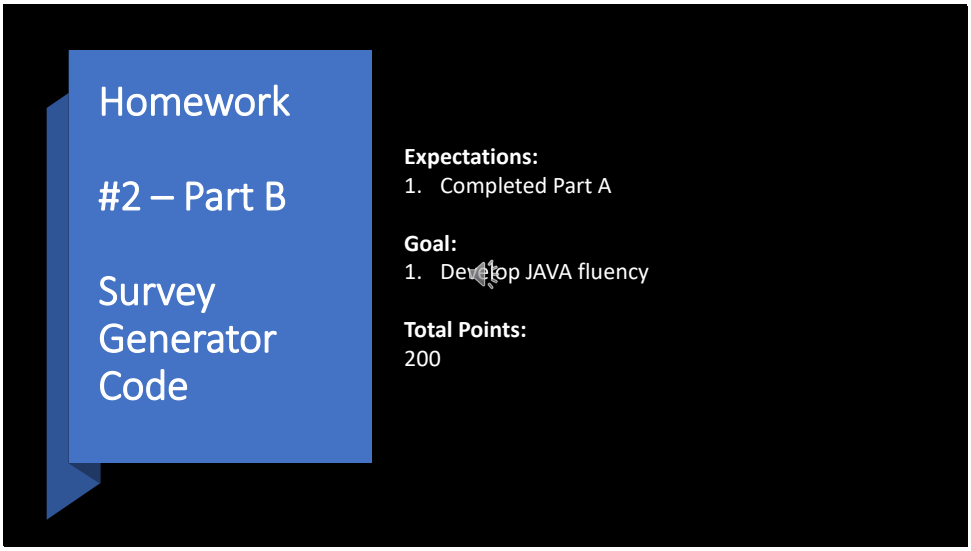
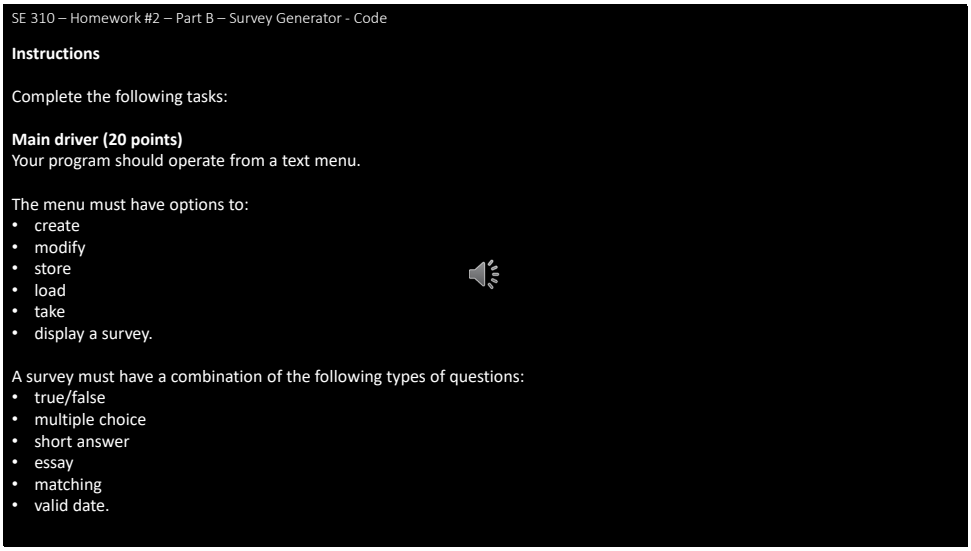
Slide 1



For Part B of your second homework assignment, you are going to code your design for a survey system.
Therefore, you obviously must have completed Part A.
However, you may change any aspect of your design from Part A when completing Part B.
The goal of this assignment is to elevate your mastery of JAVA.
Note that the assignment is graded out of 200 points.

Slide 2



The following are the instructions for this assignment.

Please complete the following tasks:

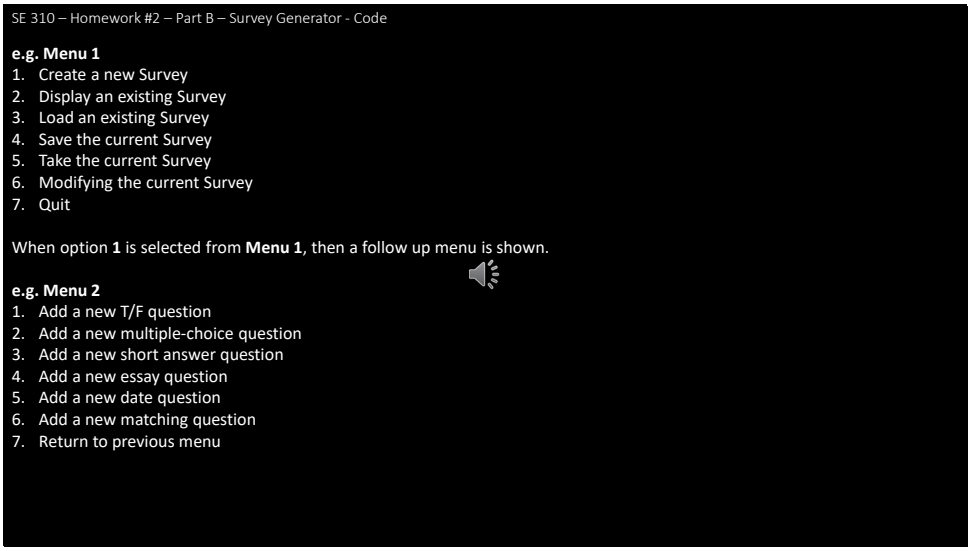
First, the main driver (20 points)
Your program should operate from a text menu.

The menu must have options to:

- create
- modify
- store
- load
- take
- and display a survey.

A survey must have a combination of the following types of questions: true/false, multiple choice, short answer, essay, matching, and valid date.

Slide 3



Let's look at what some example menus would look like:

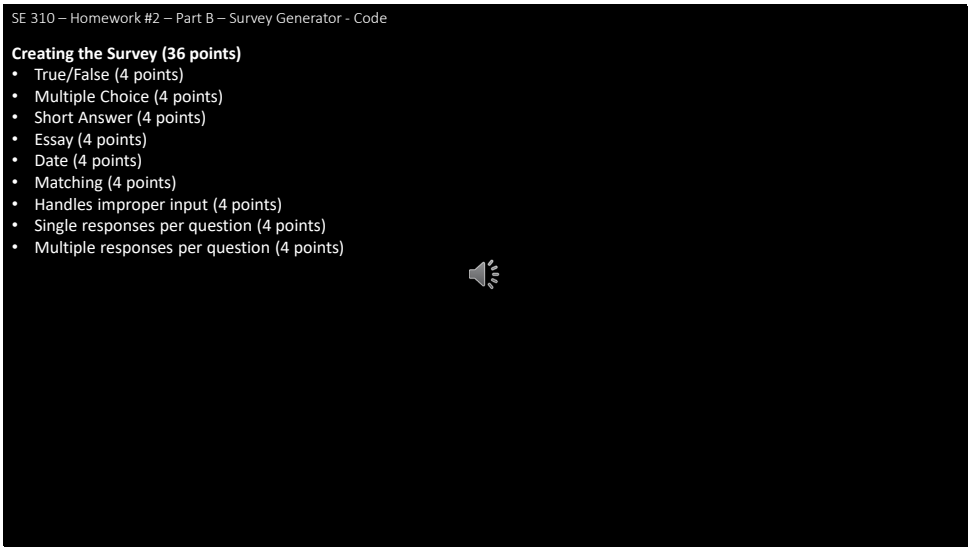
For Example, Menu 1 could contain

- Create a new Survey
- Display an existing Survey
- Load an existing Survey
- Save the current Survey
- Take the current Survey
- Modifying the current Survey
- Quit

When option **1** is selected from **Menu 1**, then a follow up menu is shown.

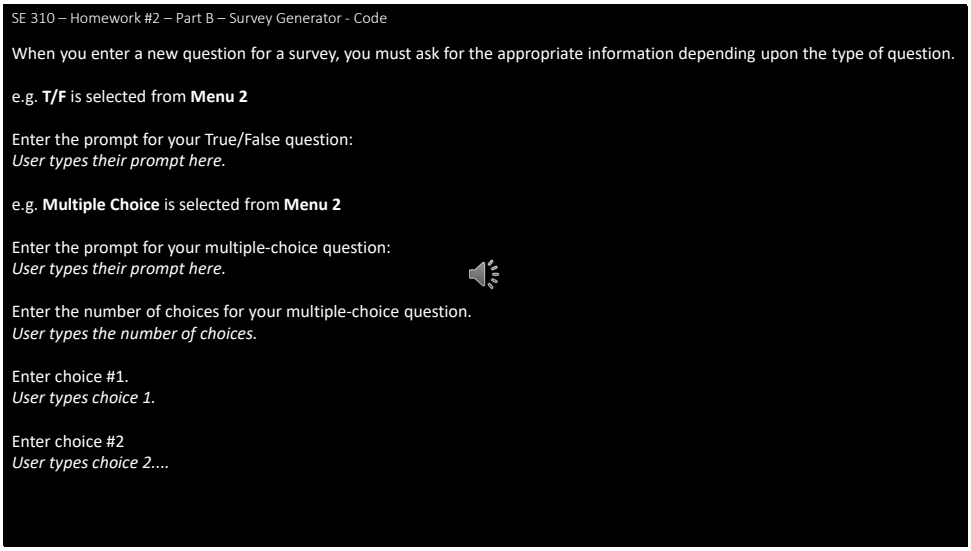
- Add a new T/F question
- Add a new multiple-choice question
- Add a new short answer question
- Add a new essay question
- Add a new date question
- Add a new matching question
- Return to previous menu

Slide 4



The next section is for Creating the Survey which is worth 36 points.
Breaking it down further each section for creating a survey is worth 4 points.
The sections include:
True/False
Multiple Choice
Short Answer
Essay
Date
Matching
Handles improper input
Single responses per question
Multiple responses per question

Slide 5



When you enter a new question for a survey, you must ask for the appropriate information depending upon the type of question.

For example if **T/F** is selected from **Menu 2** you could have the prompt

Enter the prompt for your True/False question:
User types their prompt here.

If **Multiple Choice** is selected from **Menu 2** you could have the prompt

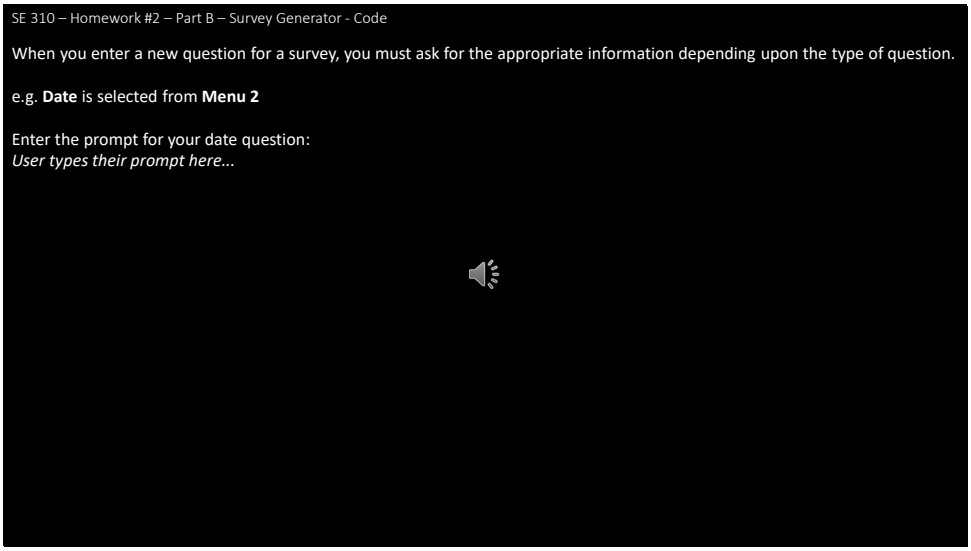
Enter the prompt for your multiple-choice question:
User types their prompt here.

Enter the number of choices for your multiple-choice question.
User types the number of choices.

The you prompt, Enter choice #1.
User types choice 1.

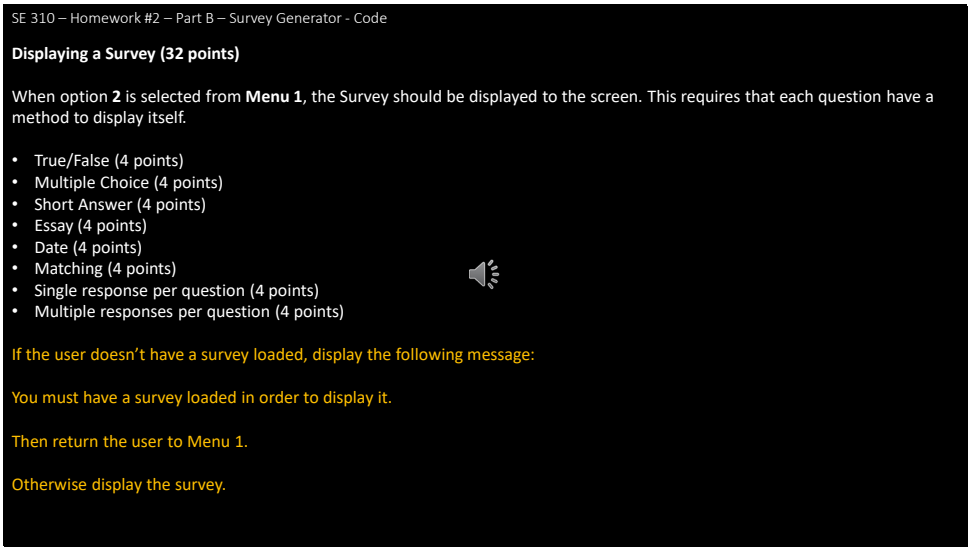
Then you prompt, Enter choice #2
User types choice 2....

Slide 6



If **Date** is selected from **Menu 2**
you could have the prompt,
Enter the prompt for your date question:
and then the
User types their prompt here...

Slide 7



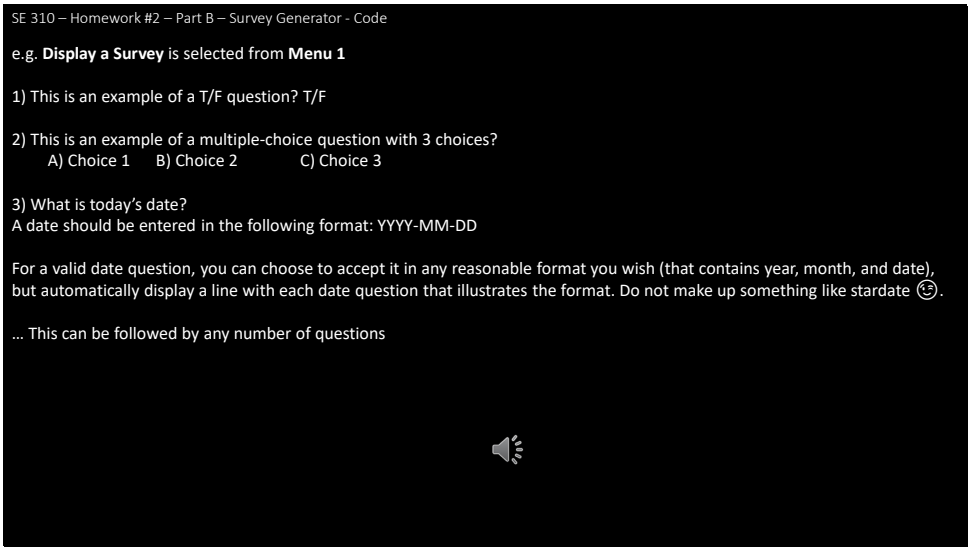
The next section is for Displaying a Survey which is worth 32 points.

When option **2** is selected from **Menu 1**, the Survey should be displayed to the screen. This requires that each question have a method to display itself.

Each part is worth 4 points.
The parts include:

True/False
Multiple Choice
Short Answer
Essay
Date
Matching
Single response per question
Multiple responses per question

Slide 8



For example, if **Display a Survey** is selected from **Menu 1** the entire survey is displayed to the screen.

It could look as follows:

Question 1)
This is an example of a T/F question? T/F

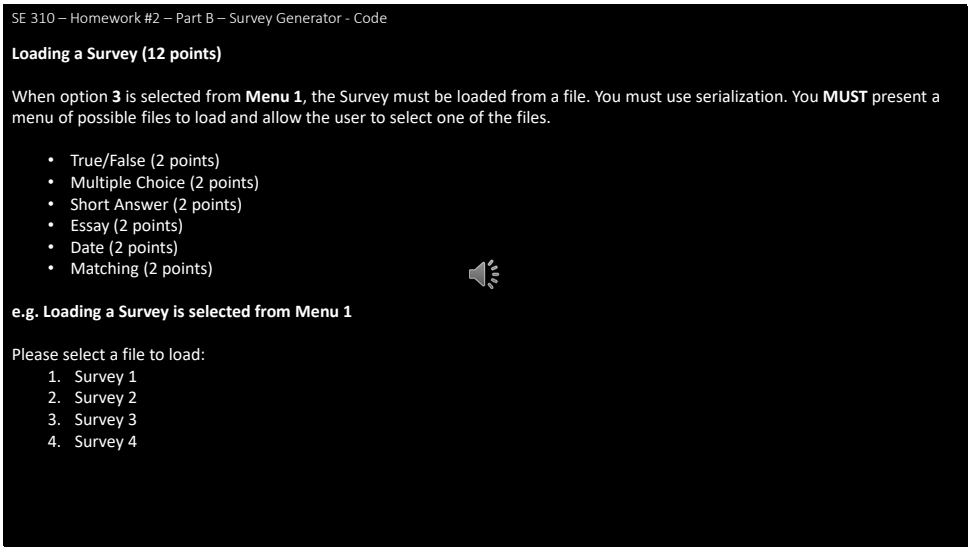
Question 2)
This is an example of a multiple-choice question with 3 choices?
Choice 1 B) Choice 2 C) Choice 3

Question 3) What is today's date?
A date should be entered in the following format: YYYY-MM-DD

For a valid date question, you can choose to accept it in any reasonable format you wish (that contains year, month, and date), but automatically display a line with each date question that illustrates the format. Do not make up something like stardate 😉.

… This can be followed by any number of questions that make up the rest of the survey

Slide 9



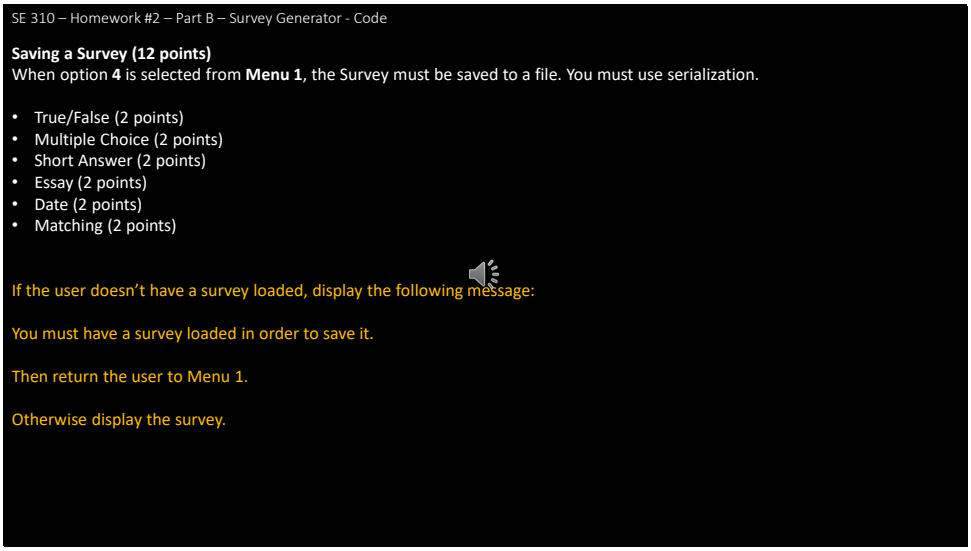
The next step is Loading a Survey which is worth 12 points

When option **3** is selected from **Menu 1**, the Survey must be loaded from a file. You must use serialization. You **MUST** present a menu of possible files to load and allow the user to select one of the files by entering an index or number of the file.

Each question type is worth 2 points. This includes
True/False
Multiple Choice
Short Answer
Essay
Date
Matching

For example, Loading a Survey is selected from Menu 1
and the prompt
Please select a file to load:
is displayed. Then the program queries the directory and loads each survey found with an index for easy selection.

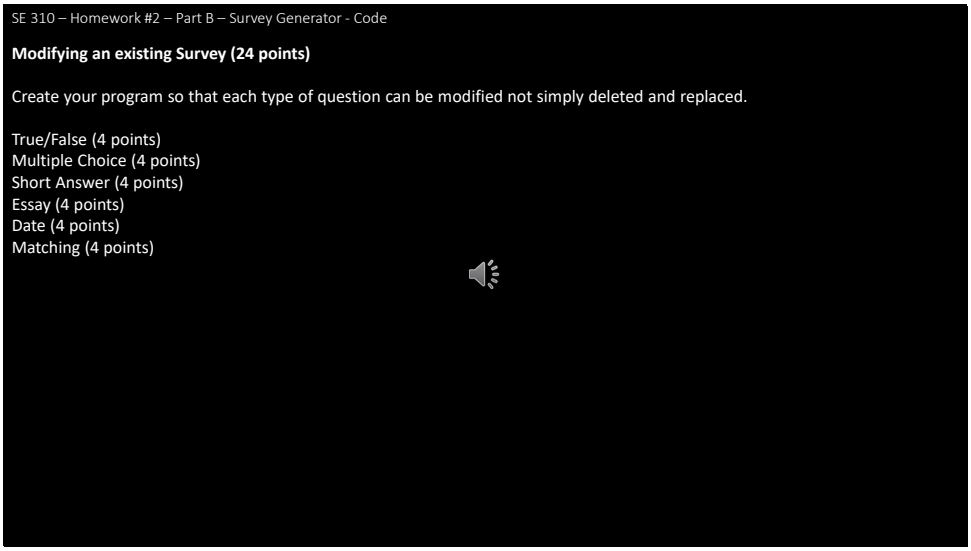
Slide 10



The next section is Saving a Survey which is worth 12 points.

When option **4** is selected from **Menu 1**, the Survey must be saved to a file. You must use serialization.
Saving each type of question is worth 2 points.

Slide 11

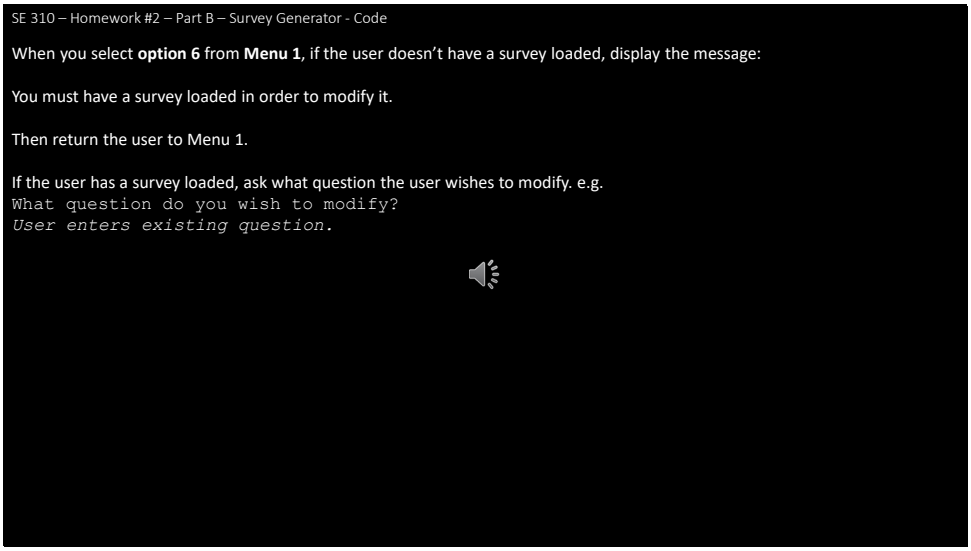


The next section is modifying an existing Survey which is worth 24 points

Create your program so that each type of question can be modified not simply deleted and replaced.

Do this for each of the six types of questions.

Slide 12



Modifying a question works as follows:

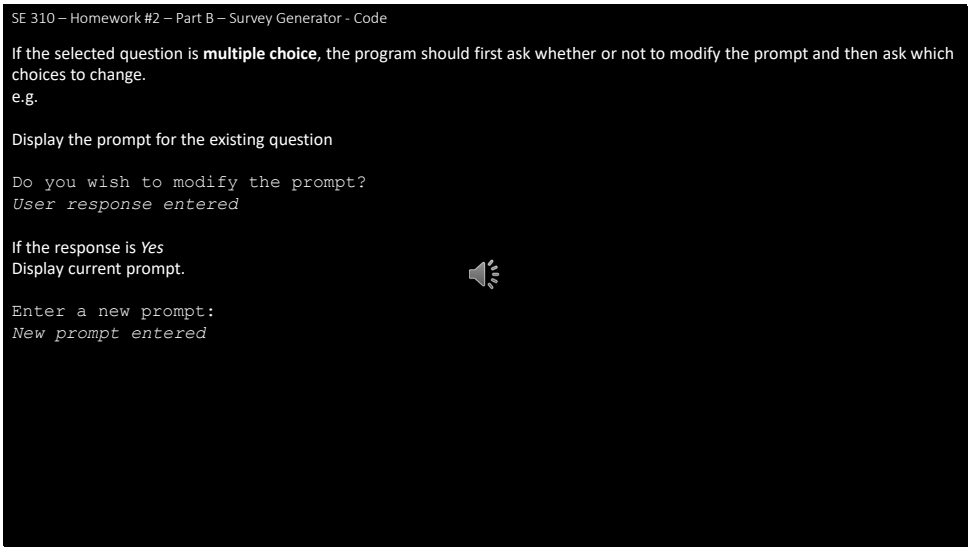
When you select **option 6** from **Menu 1**, if the user doesn't have a survey loaded, display the message:

You must have a survey loaded in order to modify it.

Then return the user to Menu 1.

If the user has a survey loaded, ask what question the user wishes to modify.
For example you could prompt the user with
What question do you wish to modify?
and then allow the user to enter the number of the desired question.

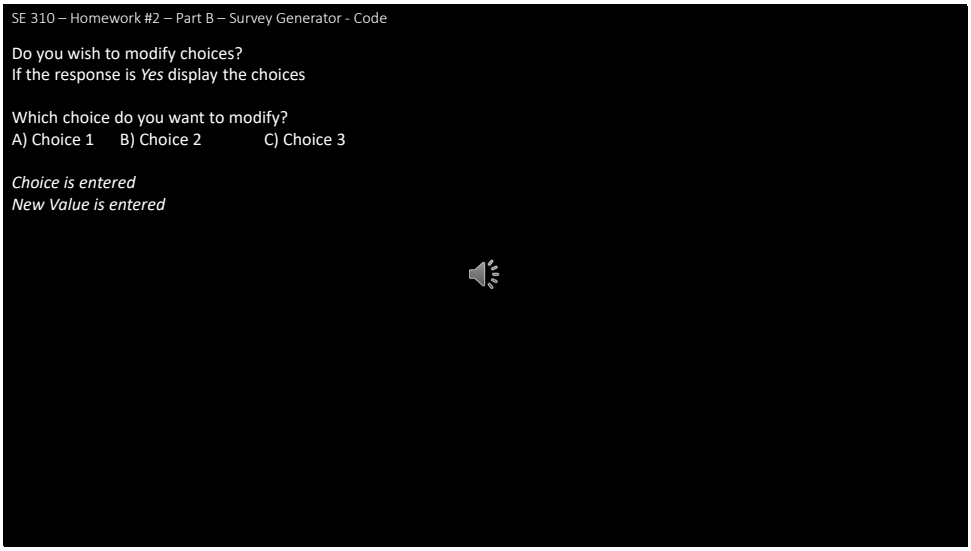
Slide 13



If the selected question is **multiple choice**, the program should first ask whether or not to modify the prompt and then ask which choices to change.
For example you could display the prompt for the existing question
And then ask
Do you wish to modify the prompt?
To which the user could reply Yes or No.

If the response is *Yes*
You would display current prompt.
and then display the prompt
Enter a new prompt:
and allow the New prompt to be entered

Slide 14



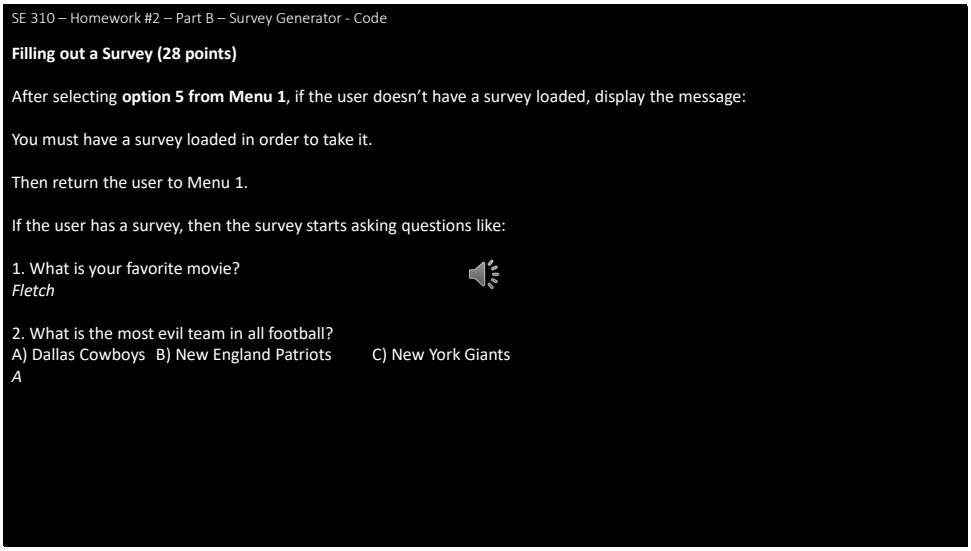
Next you can present the prompt,
Do you wish to modify choices? and allow the user to enter Yes or No.
If the response is *Yes* display the choices and a prompt like

Which choice do you want to modify?
and then list the choices.

The user enters a new choice and a new value for the choice.

This may seem like overkill, but think about what would happen if you just destroyed the question and started over.
If there were a complex prompt or many choices, you'd be repeating a lot of work.
Also, imagine if instead of text we had large videos for prompts and choices.
Would you want to reload all those files if all you needed to do was replace one of them?
Probably not.

Slide 15



The next section is filling out a Survey which is worth 28 points.

After selecting **option 5 from Menu 1**, if the user doesn't have a survey loaded, display the message:

You must have a survey loaded in order to take it.

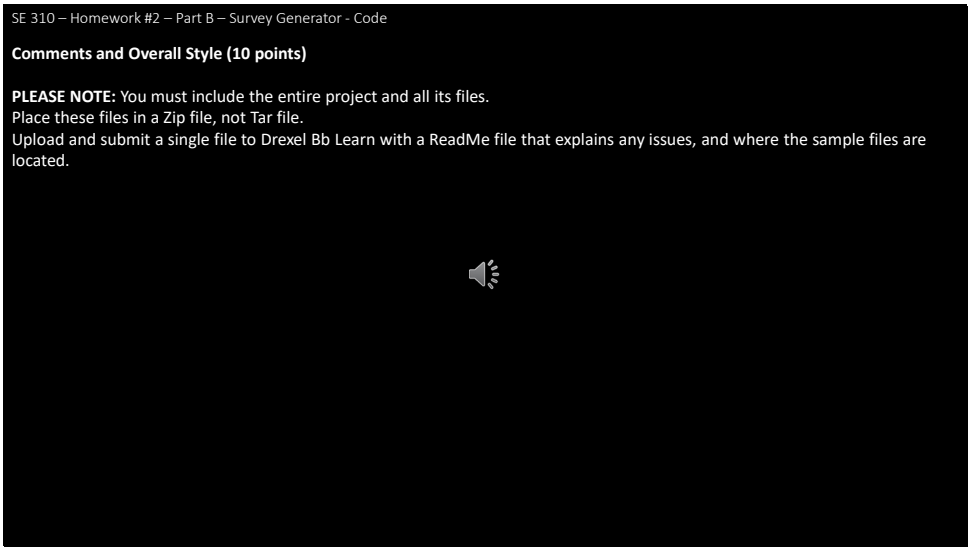
Then return the user to Menu 1.

If the user has a survey, then the survey starts asking questions like:

1. What is your favorite movie?
The user might respond, Fletch

Then the program asks question 2.
What is the most evil team in all football?
A) Dallas Cowboys B) New England Patriots C) New York Giants
The user could respond A.

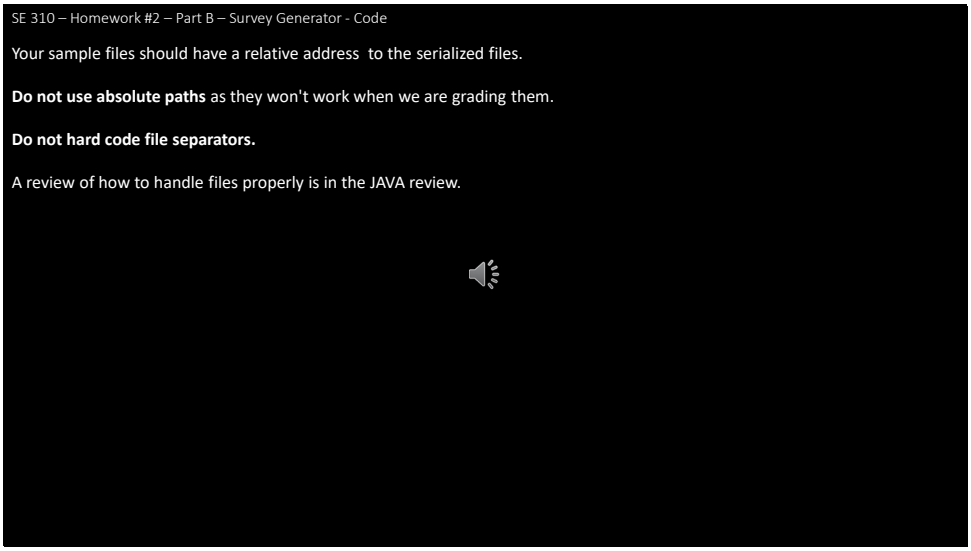
Slide 16



The next section is your Comments and Overall Style of coding and they are worth 10 points

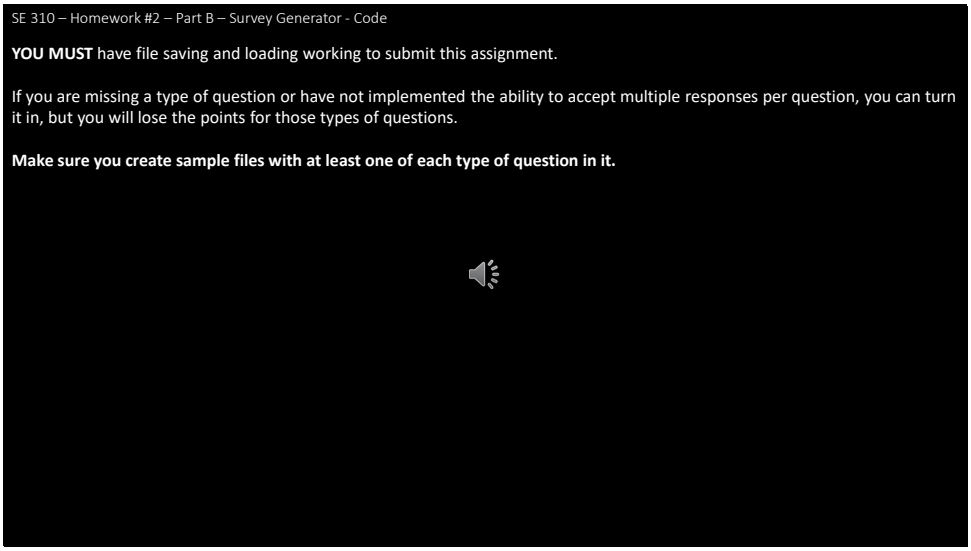
PLEASE NOTE: You must include the entire project and all its files.
Place these files in a Zip file, not Tar file.
Upload and submit a single file to Drexel Bb Learn with a ReadMe file that explains any issues, and where the sample files are located.

Slide 17



Also, your sample files should have a relative address to the serialized files.
Do not use absolute paths as they won't work when we are grading them.
Do not hard code file separators.
A review of how to handle files properly is in the JAVA review.

Slide 18

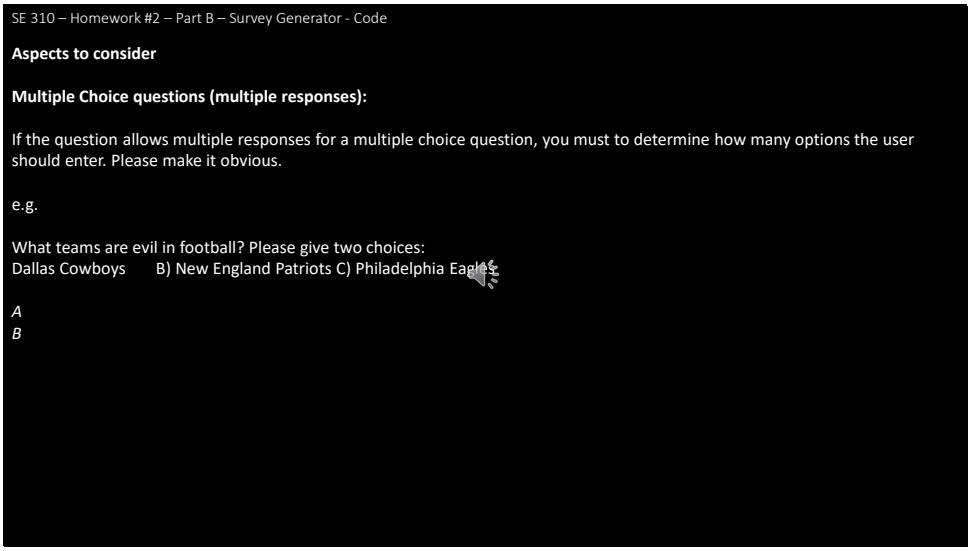


YOU MUST have file saving and loading working to submit this assignment.

If you are missing a type of question or have not implemented the ability to accept multiple responses per question, you can turn it in, but you will lose the points for those types of questions.

Finally, make sure you create sample files with at least one of each type of question in it.

Slide 19



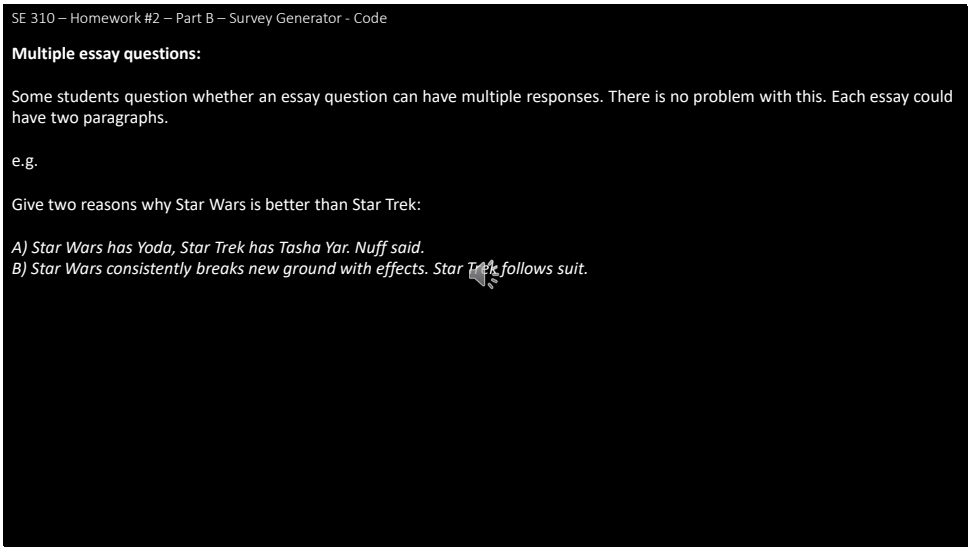
Here are a few additional aspects to consider

Some Multiple Choice questions allow multiple responses:

If the question allows multiple responses for a multiple choice question, you must to determine how many options the user should enter. Please make it obvious.

For example if you question is
What teams are evil in football? You should make part of the prompt, Please give two choices:
Then list the choices.
When the person responds to a survey they would then enter two choices, each on a different line.
and of course, the real correct response to the most evil teams in football are both the Cowboys and the Patriots.

Slide 20

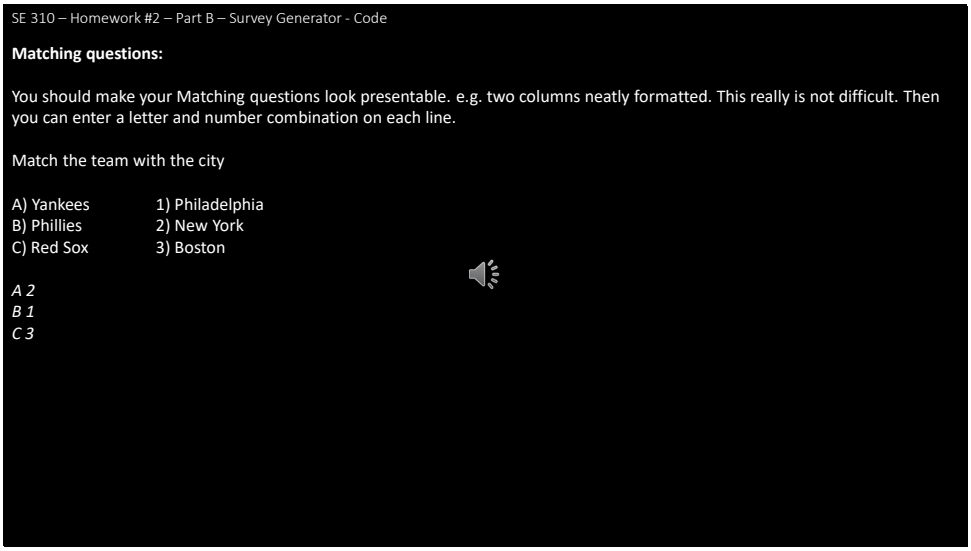


Students often get confused on how an essay question can have multiple responses.

There is no problem with this. Each essay could have two paragraphs.

For example
You could create a prompt like
Give two reasons why Star Wars is better than Star Trek:
Then have two responses like:
A) Star Wars has Yoda, Star Trek has Tasha Yar. Nuff said.
B) Star Wars consistently breaks new ground with effects. Star Trek follows suit.

Slide 21

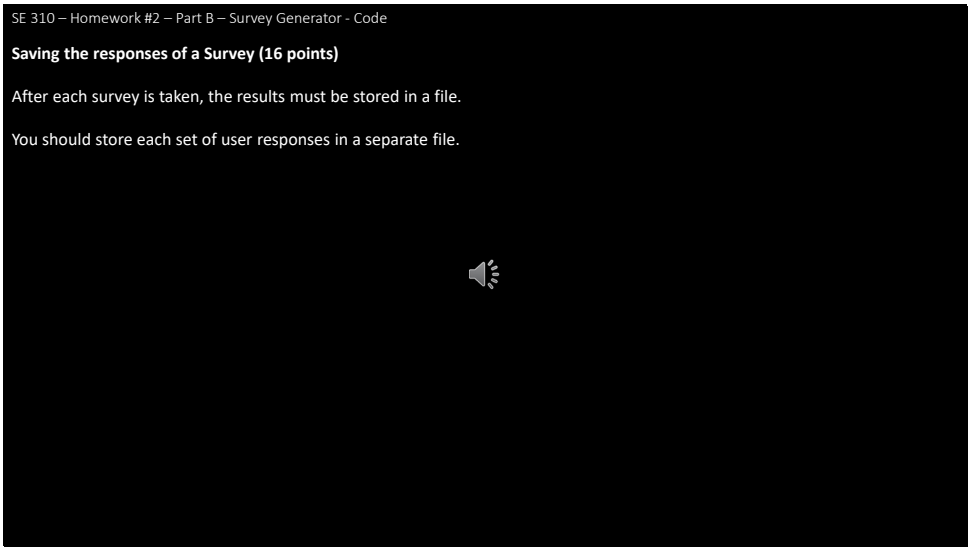


A note on Matching questions:

You should make your Matching questions look presentable. i.e. two columns neatly formatted.
This really is not difficult.
Then you can enter a letter and number combination on each line.

For example you can ask the question to
Match the team with the city
and provide two lists of teams
Then the user could respond
A 2 on one line
B 1 on another line
and finally C 3

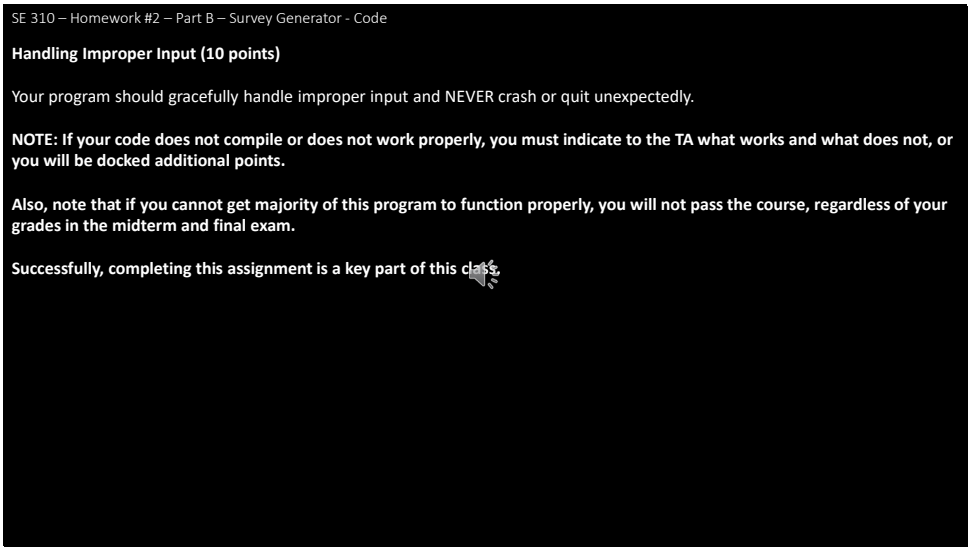
Slide 22



The next section is saving the responses of a Survey which is worth 16 points.

After each survey is taken, the results must be stored in a file. You should store each set of user responses in a separate file.

Slide 23



Just because your program appears to work doesn't mean you are done.
You have to handling improper Input which is worth 10 points.

Your program should gracefully handle improper input and NEVER crash or quit unexpectedly.

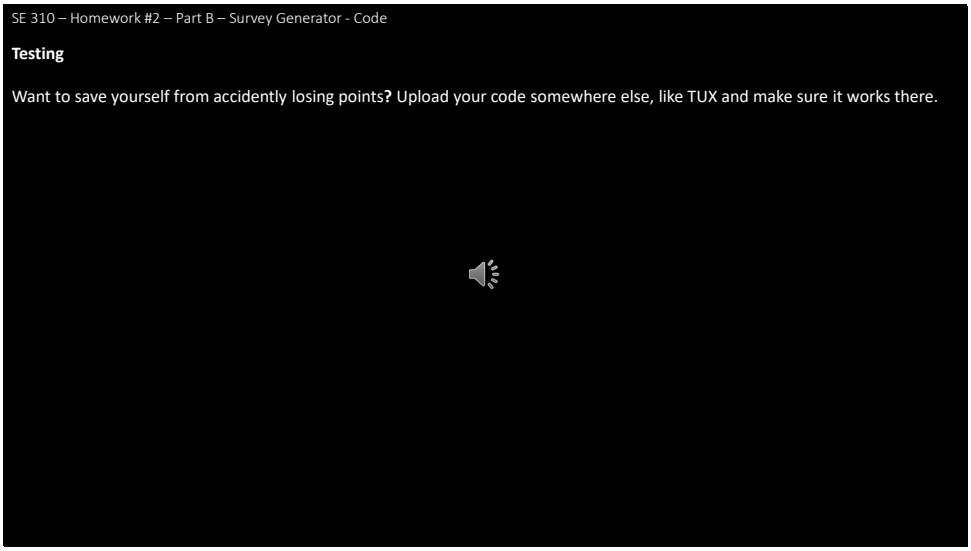
NOTE: If your code does not compile or does not work properly, you must indicate to the TA what works and what does not, or you will be docked additional points.

We want you to respect the TA's time and having them try things you know does not work is not doing so.

Also, note that if you cannot get the majority of this program to function properly, you will not pass the course, regardless of your grades in the midterm and final exam.

Successfully, completing this assignment is a key part of this class.

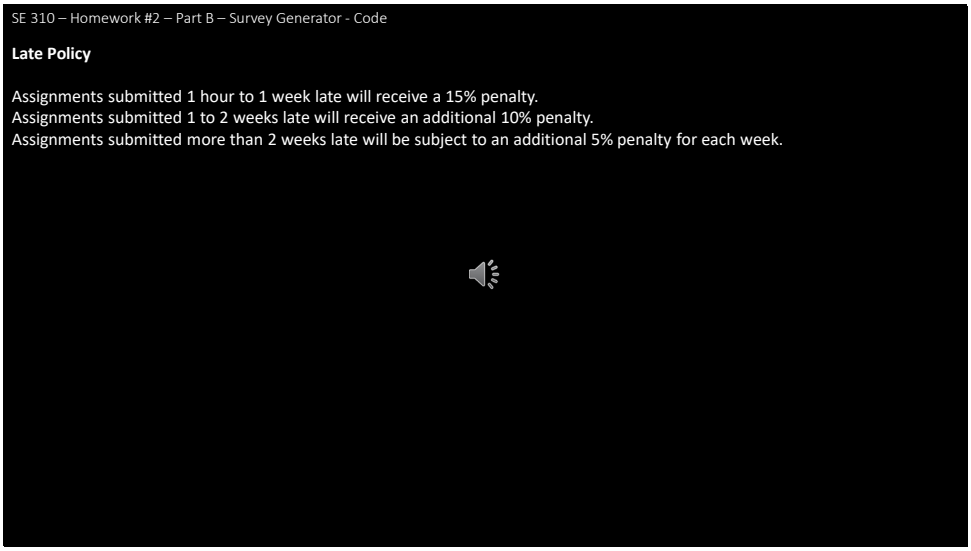
Slide 24



While you do not get explicit points for testing, save yourself from accidently losing points**.**

Upload your code somewhere else, like TUX and make sure it works there.

Slide 25



Our Late Policy is as follows.

Assignments submitted 1 hour to 1 week late will receive a 15% penalty.
Assignments submitted 1 to 2 weeks late will receive an additional 10% penalty.
Assignments submitted more than 2 weeks late will be subject to an additional 5% penalty for each week.

Please be aware though, falling behind in this class creates huge problems as one assignment builds on the others.