Yel - Blue

Yannie - Red

Darla - Yellow

Remarks - Green

Yel: Good day! I am Mariel Tibio, the Lead Programmer. Here with me are Programmer 1, Darla David, and Programmer 2, Marianne Buama. We are Git Git Aww and we will present our DSA Finals machine problem.

#### -- Contents of the File --

Yel: Before running the program, we will first show the contents of our text files. We made three text files named customer movie, customer name, and customer rent. \*\*show customer text file\*\* The customer movie text file shows a list of the 25 movies we have prepared. 5 movies for each of the following genre: Horror, Romance, Scifi, Action, and Comedy. Each movie has its own ID, name, genre, production, number of copies, and poster filename. \*\*show customer name\*\* Meanwhile, the customer name text file shows the ten existing customers including their ID, name, and address. And lastly, the customer rent text file shows nothing because no one has rented a movie yet.64

#### -- [5] Display all Videos --

Darla: Now, let’s run the program. The first thing we’ll do is to display all the movies that were loaded from the customer movie text file to its designated linked list. As you can see here, the ID, name, genre, production, and number of copies of all the movies were displayed. \*\*choice 5-make sure full screen\*\*

#### -- How data are transferred from a text file to the data structure --

Yel: This was made possible by implementing a loop in the program where it will read each line of the text file and assign it to an element in a node of a linked list. For example, \*\*show customer text file\*\* the first line of the text file will be read and will be assigned to the movie ID element in a node then the next line will be read and will be assigned to the second element in a node as the movie title. This will go on until it reaches the fifth line wherein the fifth line will be assigned an element in a node as the movie poster filename. After the fifth line, the linked list finally has its first node and this will continue until it reaches the end of the text file, making 25 nodes. Same logic was used in transferring our other text files in their designated linked lists. We would like to note that we used three linked lists which are for the movies, customers, and the customers who rented respectively.

#### -- [7] Customer Maintenance, [1] Add New Customer, [2] Show Customer Details --

Darla: Going back to the program, \*\*customer maintenance\*\* we will now go to the menu where we can control the customers. \*\*add new customer\*\* We will add a new customer named Sailor Moon and her address is Binan, Laguna. After adding the new customer, it will be added as a node along with the existing nodes in the customers linked list. \*\*show customer details\*\* Picking choice 2 in the customer maintenance menu will show the details of any customer including the new customer, Sailor Moon \*\*input 11 in customer ID\*\*.

#### -- [1] New Video --

Darla: If we want to add a movie, we may do so by entering 1 as our choice. \*\*choice 1\*\* The new movie we’re going to add will have an auto-generated ID and its details are Rio as title, Comedy as genre, Blue Sky as production, 5 as number of copies, and rio\_comedy.jpg as poster filename.

#### -- [2] Rent a Video --

Darla: The program will go back to the main menu and this time we are going to rent a movie. \*\*choice 2\*\* We are going to make customer 11, Sailor Moon, rent the new movie. \*\*Enter 11 and 126\*\* After inputting the movie ID she wants the rent, the details of the movie will be shown and the number of copies will be updated. We are going to make Sailor Moon rent another movie. This time an existing movie which will be used later.

#### -- How quantity are deducted after renting --

Yel: The quantity of the copies are deducted after renting by first matching the movie ID the user wants to rent to the movie ID in a node. If the node is found, the element assigned as the number of copies will be decremented.

#### -- [6] Check Video Availability --

Darla: If we want to check if a movie is available or has copies, we may do so by going to check movie availability \*\*choice 6\*\*. If we want to check the availability of the movie Sailor Moon rented, we just enter the movie ID. Even though the movie, Rio, was rented earlier, its status will be seen as available because there are still copies left. It will also show the poster of the movie.

#### -- How image is displayed --

Yel: The poster of the movie is displayed by first matching the movie ID the user wants, to check the movie ID in a node. If the node is found, the element assigned as the filename of the poster will be called.

#### -- [7] Customer Maintenance, [3] List of Videos Rented by a Customer --

Yannie: Now, we will show you the list of movies rented by a customer. In this example, we will use Sailor Moon’s list of rented movies. \*\*choice 7-choice 3\*\* As you can see here, the movies she rented earlier were shown including its movie ID and title.

#### -- [3] Return a Video --

Yannie: If we want Sailor Moon to return a movie, we enter 3 as our choice in the main menu. \*\*choice 3\*\* After inputting Sailor Moon’s customer ID and the movie ID she wants to return, the details of the movie will be shown and the number of copies will be updated.

#### -- How quantity are updated after returning --

Yel: The quantity of the copies are updated after returning, by first matching the movie ID the user wants to rent, to the movie ID in a node. If the node is found, the element assigned as the number of copies will be incremented.

#### 8-- [4] Show Video Details --

Yannie: We will now show the video details of the movie returned by Sailor Moon. \*\*choice 4-126\*\* We will input the movie ID and it will show the details of the movie and its poster.

#### -- [8] Exit Program --

Yannie: If we are finished with the program, we will now exit the program by choosing 8. \*\*choice 8\*\* The program will save all the updated information of the linked lists to the text file as it exits.

#### -- Transferring of data from data structures back to the text file --

Yel: This was made possible by implementing a loop in the program where it will read every node in the linked lists and write each element of the node in their designated text files.

#### -- Contents of the File --

Yel: This can be proven by showing you the updated contents of the text file. \*\*show movie text file\*\* The new movie is now in the customer movie text file. \*\*show customer name file\*\* The new customer is now in the customer name text file. \*\*show customer rent file\*\* And lastly, the movie that was rented but wasn’t returned will appear in the customer rent text file.

#### -- Any inclusions of error handling --

Yel: Before ending our demonstration, we would like to show one of the error handling the program can do. Let’s say we want to display the movie details of a movie that was never in the list. \*choice 4-128\*\* It will output an error saying the movie is not in the list and the movie poster is not found.

Yannie: And that is all for our machine problem program demonstration. Thank you!