Prathyoosha Chaya

CS 163 DATA STRUCTURES

Prof. Karla Fant

**DESIGN WRITE-UP 2:**

**HEADER FILES:**

The two header files will include the game class, stack class, and node struct definition, with the function prototypes of the following public member functions.

GAME HEADER FILE: This header file includes the game class information along with the node structure, card data structure (subject, question, level, and description). Its private members include a pointer to the front of the LLL, and a counter for the number of cards.

Constructor: The purpose of the constructor function is to initialize the data members. Here, the pointer to the front of the LLL will be set to null, and the number of cards is set to 0.

Destructor: The destructor will call the write function, then deallocate all previously allocated dynamic memory - the pointer to the LLL (the dynamic arrays within the card node data will be deallocated in the node destructor).

Setup function: The public setup function first creates an array to store the pointers to each data member using the address-of operator. Then, using the algorithm for shuffling the data in the assignment, it picks a random index number in this array of pointers and dereferences it to access the card info. It then pushes this into the stack LLL of arrays by calling the push function and setting the pointer to null. If the pointer has already been set to null, then it goes to the next index in the array of pointers until it finds one that has not been pushed already. This function takes no parameters.

Add card function: The add card function is public and takes no parameters. It returns an integer value depending on the case of the success/failure. It will add the card details to the next index in the LLL, and if the array is full, it will create another node. This function takes the card details to add by parameter and returns the integer value of the success or failure case.

Remove card function: The add card function is public and first displays the questions on all the cards along with a number corresponding to the card. The user selects the number of the card they would like to remove, which is then traversed to and removed. A trailing previous pointer joins the LL after the removal. It returns

Display all function: This public function will display all of the cards in the LLL by traversing through the list and displaying every card with its data. It returns the success/failure case (if the list is empty) by Boolean value, and it does not have any parameters.

Play function: The public play function will be called after the setup function. It will prompt the user in main to choose if they want to draw a card. If they want to draw one or peek at the next card. Their choice is sent as a parameter to the play function. If they would like to draw a card, then the pop function in the stack class is called. If the user would like to instead peek at the next card, before drawing it, it will call the peek function in the stack class. The user enters if they would like to continue or not, and the result is returned back to main as a Boolean value. If they said yes, then the function is called again.

Write function (to accompany load function): The purpose of this function is to write the contents of the LLL of arrays back into the file. It has no parameters. The card data will be outputted to the file (no append), separated with “;” delimiters. At the end of the subject, a “\n” delimiter is placed and the index is incremented. This loops until the index meets the card counter value.

Load function: The purpose of the load function is to take the card information in the text file and store it into a LLL of arrays. It uses a pointer to the array of structures and an array index counter, which are the class’s private data members, and uses the opened text file to write the data. It uses the existing delimiters mentioned in the “write” function to read into each card’s fields.

CARD STACK HEADER FILE: Includes public member functions and private pointer to LLL of Arrays, and pointer of type card to point to a dynamic array of pointers to each card.

Push function: The public push function will push the data from the game class LLL of arrays into the stack LLL of arrays. It is called within the setup function and returns an integer value of the current index. Its parameters are the original dereferenced card that the pointer in the setup function pointed to and the current index.

Pop function: The public pop function will pop the first card from the stack LLL of arrays if called by the play function. It uses the modulus operator to access the specific array member, then increments the index. This function does not take any parameters, and returns a Boolean value of the success or failure.

Peek function: This public function can be called by the play function to display the uppermost item in the stack. It takes the current index as a parameter by value, and displays the question on that first card. It returns a Boolean value based on if the next index exists or not.

Display function: This public function will display all the cards (with the subject, question, level, and description) in the deck in their randomly generated order, as stored in the stack. There are no parameters. It will not use the array of pointers to display, but instead go through the LLL of arrays card by card to display. It returns a Boolean value of a success or failure (empty stack). It uses the incremented index number divided by the array size to find the number of traversals required to display each item, and uses the incremented index modulus the array size to get the index.

IMPLEMENTATION FILE:

The first implementation file will contain the function definitions below.

In the main function, the random number generator is seeded, then a menu interface will be implemented for the user to interact with the game. An object of type game “trivia”, and an object of type card stack “deck” is created. Then, the input file and output file variables (ifstream/ofstream type) are created, and a “menu option” variable (type int, to keep track of the menu item entered). The text file is opened, and the load function is called.

Then, the menu for the game class functions is outputted by calling the menu function with parameter “true”. The user can choose which option they would like to execute from the following menu format:

1: Display all cards.

2: Add a new card.

3: Remove a card.

4: Play!

5: Save cards and quit game.

6: Quit game without saving cards.

The user will be prompted to enter a number, which is stored into the “menu option” variable once the function exits. If a 1 is entered, then the “display” function is entered. At this point, all data from the text file will have been read into the. Once the function is exited, the menu function is entered again. Once “play” is selected, the menu function runs again with the parameter as “false” to output the stack class functions menu. The functions continue to be called until the user decides to quit (or save and quit).

1: Draw a card.

2: Save and quit game.

3: Quit game.

Menu function: This function takes 1 bool parameter by value (to display either the game class-related menu or the stack class-related menu) and returns an int value. It outputs the menu for the user to select, and returns the user’s selection. The menus displayed are shown in the main function.