Prathyoosha Chaya

CS 202 PROGRAMMING SYSTEMS

Prof. Karla Fant

**DESIGN WRITE-UP 1:**

This program will attempt to have the following self-regulating classes in order to act in an object-oriented fashion:

The CS game class is responsible for managing the game progress, and will include the private queue data structure, along with function definitions of the public member functions dealing with queue management (queue player, dequeue player), and general game setup/ending methods.

Setup function: This function will be called at the start and will set up the game. It will first call the build map function in the maze class, then set up the player groups. It will call the add player function in the player class until there are at least 2 players in each group. It returns a success/failure flag.

Queue player function: This function will queue a player to the end of the queue once their turn has finished. If the player has reached the endpoint, then they will not be queued, their information will be entered in the “won” player array of size 2, and the number of players in the game will decrease by 1. If this number has decreased by 2 from the starting player number, then the end game function is called. The function returns an integer value of the player number queued.

Dequeue player function: The dequeue function dequeues a player from the queue once they have finished their turn. It is then followed by the queue function, to add the player to the end of the queue.

End game function: This function will execute if the number of players in the game has decreased by 2 players from the starting number. The 2 players that are placed in the “won” array will be displayed as the two winning players, then the game finishes.

The map class is responsible for managing the maze, and will include the private maze graph, which is a pointer to a collection of checkpoints. This class “has a” checkpoint class. Along with function definitions of the public member functions dealing with graph management (build map, set the start and ending checkpoints).

Build map function: The build map function will create a randomly generated maze with the following considerations: each checkpoint will have a maximum of 3 connections, there are at most 20 checkpoints, and there will be a maximum of 5 dead ends. This will be used by the setup function in order to build the maze before the game begins.

Assign start function: The assign start function will pick a random checkpoint to be the starting checkpoint for the maze. It is called by the setup function in the game class.

Assign end function: The assign end function is called after the assign start function in the setup function of the game class.

The checkpoint class will include the private checkpoint information (checkpoint number, checkpoint name, and checkpoint description). The checkpoint class is contained by the map class.

The location class will include the private location name and description, along with function definitions of the public member functions to display the location information, and change the location for the player. It contains the public display info function to display the name and description of the current location, as well as a change location function to change the player’s current location. It is a base class to the player class.

The player class is responsible for managing the player information, and will include the private list of players, the player number, group number, name, and list of friends. The public functions are to add a new player (called by the setup function), and to assign a reward to a player after calling the spin function in the reward class. The player “has a” reward and friend list. The reward class is contained in the player class, and contains a public spin function to select a random reward after each turn. The set reward player function will call this reward class function. The friends list class is also contained in the player class, and is responsible for managing the friends list. It has public functions to group the players together, and to display the groups in the game.

Constructors: The purpose of the constructors are to initialize the data members. All classes will have default constructors, along with copy constructors in the case of dynamic memory. Initialization lists will be used where possible.

Destructors: The destructors will deallocate all previously allocated dynamic memory in their respective classes.

Convert to uppercase function: The purpose of this private function is to convert the inputted array of characters to uppercase, using the toupper functionality in the cctype library. It takes a source and destination array of characters as input parameters, and changes the contents of destination to be the uppercase version of the source array of characters, going character by character. This will be a puclic method in the base class (location class).

IMPLEMENTATION FILES:

The main implementation file will contain the function definitions below. Others will contain the function definitions for the class methods specified above.

In the main function, the menu interface will be implemented for the players to interact with the game. An object of type game, and an object of type map are created for the purpose of calling game and map class functions and passing the user’s information to be added into these functions. Then, a “menu option” variable (type int, to keep track of the menu item entered). The setup function for the game class is called, and players are entered, followed by the build map function.

Then, the game menu 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:

Player # Actions:

1: Move

2: Use a reward

3: Go to start

4: Retrieve information.

5: Quit game

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 player can change their current checkpoint to any surrounding checkpoint. Once the function is exited, the menu function is entered again. The functions continue to be called until the user decides to quit (or save and quit). The program can be quit without writing to the file, or quit after writing the contents of the table to the file.

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 menu displayed is shown in the main function.

