**RED TAG**

A PROGRAM FOR LEAD ROTATIONS

Author: Nathan Abdelmalek

**General Overview**

The Red Tag program is designed to help Hersheypark ride leads create easy to use rotations, and to notify them about potential staffing challenges. These two tasks are ones that ride leads, especially newer ride leads, often struggle with. This leads to frustrations with ride operators, and likely costs Hershey Entertainment and Resorts thousands of dollars each year, in the form of child labor law violations. Given an accurate input of ride and operator specifications, this program can produce a fair rotation that can adapt to changes in staffing throughout the day. It runs off a simple, easy to understand, command line interface that allows for minimal distractions. Additionally, the program can identify what operators are minors, when they need their breaks, and whether there are currently enough extras to cover those breaks. It then puts these breaks into the rotation if it is possible to do so. Ultimately, a ride lead following this program completely should have no issues related to minor violations or breaks.

**Inputs**

The program depends on reliable user input prior to any rotations being created. Failure to enter accurate data during startup will require the user to restart the program. The operator must first enter the name of the ride they are working on. If the ride is a recognized configuration, the program will give you the option to load the data automatically. Currently, all north lead rides are recognized. If the name is not recognized (south or central lead ride, typo, etc.) the user must enter the data manually. This first involves entering the minimum amount of people required to operate. From there, you must enter the name of each position, whether a lead can go there, and whether a ride clerk can go there. Once you haver entered the mimum number of positions, you can also add additional positions. Notice that any positions you add now must be filled by an operator during that stage of data collection. The program will not allow you to proceed until you have filled all the positions you ave desginated. Once you have finished entering all ride data, you then proceed to park data.

The only “park data” required is the start and end times for park operational hours. They are required questions (you must answer them manually each time) and the answer must be in military time, in the format HH:MM. Failure to enter in the correct format will result in having to re-enter this data. Program cannot proceed accurately until this is completed.Finally, data about the operators must be collected. Two options exist – reading from a CSV file, or entering data manually.

If the option for CSV is used, a file name for a csv file must be provided. Failure to provide a valid CSV filename will result in the program reverting to manual data entry. The format of a valid CSV file is as follows. The first group of lines are additional positions that you want to add. Be sure that if you added the positions during the ride data collection steps, you do not add them again within the CSV file. When adding positions, the should be in the form “position name, lead status, clerk status” , where “lead status” and “clerk status” are Boolean values representing whether or not a lead or clerk can occupy this position, respectively. You may add as many positions as you want. When you are finished adding positions, enter a line with a ‘-‘ character to indicate you are now switching over to employees. (If there are no additional positions to add the ‘-‘ character will be on line 1 of the file). From there, each line will be interpreted as a new employee. Each line should be in the form “operator name, age, shift start, shift end”. The name of the lead MUST be “Lead,” or the program will fail. Additionally, operator names cannot contain spaces (“ “). All times must be in military time in the format HH:MM. If there are more operator names than positions, additional operators will simply be ignored. If there are more positions than operators, a warning message will be given, and you will have to enter the rest of the operators information manually.

If the user opts to use manual data addition, each person (along with their age, shift start, and shift end) must be provided. As above, the name of the lead must be “Lead”, and the names of the operators cannot contain spaces, or the program will fail. Additionally, when providing data manually, an option is provided asking if the employee is working the entire day. If “yes” is selected, a shift start time of a half hour before opening, until a half hour past closing is generated. If this is not desired, please select “no” and enter the shift start and end times manually.

Once initial data entry is completed, the program enters into standard operating mode. The first rotation is created and printed without any inputs needed. No additional data entry will be required unless additional employees enter the ride.

**General Operation**

Once all data entry is complete, there are a variety of commands you can use to control the flow of the program. The commands are listed at the start of the program, along with a brief description of what they do. This menu can be accessed at any time by entering the command “h”. Failure to enter a valid command will result in a warning and another prompt for a command.

Entering the command “e” gives a full readout of the employee data in a less accessible format. This is used mostly for testing purposes, and for getting full details about all employees. The printout is in the form name-break status (have they had a break or not) – break window start, break window end- current position. Break window start/end are the earliest and latest times and employee can begin their break. For adults, this is an hour and a half after starting until an hour and a half before shift ending. For minors, this windows follows minor violation windows.

Entering the command “n” will generate the next rotation. This will print out the name of the ride and the start time for this rotation (the program assumes 45 minute rotations). It will also print out whether or not any violations are expected, and if so, when they will occur. The program will then print out the list of mandatory position and who is assigned there, and the list of optional positions and who is assigned there (if any such positions exist).

Entering the command “q” will quit the program. This will cause all data to be entered until this point to be lost. Entering the command “r” will do the same, but will automatically re-run the program from startup.

Entering the command “+” will add an employee. As above, the program will prompt you for the employees name, age, and whether or not they are working the entire day. Once they have been added, they also need a position to occupy. If, at any point in the program, you have created an optional position and the employee who once occupied it is no longer at the ride, you will not have to create the position again. For example, the program will remember that if you had one extra in the morning and they occupied the Grouper position, and they then left the ride, but you later got another extra, they will once again occupy the Grouper position. If you got an additional extra (now plus 2), you would have to enter a new position. Note that if they employee is on break and the extra position is not currently occupied, but it will be later once breaks are completed, you will have to add an additional position. Finally, the user will be prompted whether or not the employee has had a break yet. If they have already had a break, the program will not provide them with one at any point.

Entering the command “- *employee name*“ will remove an employee. If the ride is currently at mins, the program will not permit any employee to be removed. If you are at mins and wish to remove an employee, or change details regarding the employee, you must first add the employee, then remove the existing employee. If you are plus, the program will remove the employee without notification, and their data will be deleted. If they were occupying a mandatory position, an employee from an optional position will take their spot (you must enter the command “e” to see this for the current rotation, or you can view it for the next rotation by using “n”). If they were occupying an optional position, the position will be removed. Entering an operators name who dose not exist will result in no action being taken.

Entering the command “f *employee name*” will allow the user to search for an employee. If the employee is on the ride, the program will print out all available information on the employee. If the employee is not present on the ride, the program will inform the user that the person is not known.

Entering the command “br” will print the break list. This is the list of all the breaks that need to be completed, in the order that they need to be completed. This is a developer option – there is no need to use this list, as the program will automatically start and end the breaks. The list is unformatted, but it can be used for debugging purposes, or for identifying when a person will go on break.

**Sending People Home**

At the end of employee’s shifts, the program will automatically prompt the user to send them home. The program will identify if the employee is a minor, and if so, what time they will violate at. The program then presents three options for how to deal with sending someone home: acknowledging, adding a new employee to replace the person leaving, or removing an optional position.

If the user chooses to acknowledge the risk by typing the reply “acknowledge”, the program will continue with the employee in the next rotation. The program will continue to remind you about the employee every time a new rotation is generated until they are removed from the program. If the user choose to add an employee using the command “+”, the program will go through the standard procedure for adding an employee, as described above. If the user chooses to remove an optional position by entering the command “-“, the program will respond as follows. If there are no extra positions in existence, the program will remind the user of this, and prompt them to enter a new command. If there is one extra position, the program will remove this position, remove the employee that needs to leave, notify the user, and continue with the rotation. If there is more than 1 extra position, the program will allow the user to choose which extra position they would like to remove. Once the user has entered a valid selection (by entering the associated number, as prompted), the program will remove that position, remove the employee, and continue with the rotation.

**Technical Details for Developers**

This section breaks down the program on a function-by-function at a high level, allowing for developers to better understand and modify the code.

**Red Tag Classes**

The red tag classes file defines all of the classes used in the Red Tag program. This includes employee objects, which handle all the information related to employees (name, age, position, shift start/end, break status, break start/end, break window). Break window is defined as the segment of time during which an employee can go on break. For minors, this is right in the middle of the shift. For adults, the rules are much looser and have more to do with convenience. It is stored in the format [window start, window end]. There are also ride objects, which track the current state of the ride (name, mins, a dictionary that maps mandatory positions to their status lists, an optional position dictionary in the same format, the current time). A “status list” is information about a position. Each position in the dictionary is mapped to a status list in the form [current employee, lead status, clerk status]. At startup, current employee is set to None. There is also a time module that is used for tracking time throughout the day, including arithmetic and logical checking. These classes are imported at the start of the Red Tag program.

**Red Tag Command Line Interface**

The red tag CLI file handles all interaction with the user. The first prompt allows the developer to use testing mode. In testing mode, the developer can enter all data ahead of time in the testing() function. This allows for quicker runs during testing phases, without needing to manually enter data on each run. This has mostly been replaced by the ability to enter data using a CSV file, but the option is still present. Entering anything other than “test” to the first prompt will enter standard operation mode. From there, the CLI calls the information-gathering functions in order to collect the info it needs to run. Finally, the CLI enters a while loop, continually prompting the user for a command, then handling it appropriately by calling the associated command in Red Tag. If the user enters the quit command, the while loop prints the goodbye message and exits.

**Information Gathering Functions**

There are three functions used to gather three different kinds of data: ride data, park data, and employee data.

Ride data is the first function that is called. It takes no inputs, and returns a fully initialized ride object, along with an integer value for the number of positions the ride has total (mandatory and optional). The function defines a dictionary mapping ride names to set functions (explained below). It then asks the user for the name of the ride, and if it is found, presents the option to use the preset information. If the preset information is used, the setting function then handles the gathering of the data, and is returned at this point. From this point forth, the function deals only with the manual collection of information regarding the ride. The mins value is collected, then the function adds the minimum number of positions, and then asks if the user would like to add additional positions beyond mins.