Andrea Halenkamp

CSC 402

December 16, 2016

Project 4 Written Portion

For this project I decided to build something that has been an interest of mine for quite some time now. Previously this semester, I created a shiny application (<https://pokemonapp.shinyapps.io/pokemondatabase/>) in the language R using a dataset I found on a website called veekun. This project can be broken down into several steps. I did have to tamper with the dataset using R, but after saving it into a txt file I was able to use it in C++. Once I was able to load the dataset in, I had to properly read it in and save it.

The dataset I downloaded from the website, <https://veekun.com/dex/downloads>, came in the form of a SQLITE file. While I could have used a package in C++ to read the dataset, I was not interested in using all the contents of the file, which includes not only all the information on each pokemon, but also information on all other game information (such as items and maps). The data was also split into many relational tables which would have made this difficult. Instead, I processed it all through R using some code I wrote, and created one unified table with all the data I wanted to use. The columns are: id, pokemon-name, pokedex-id, first-type, second-type, base-hp, base-attack, base-defense, base-specialattack, base-specialdefense, base-speed, first-ability, second-ability, third-ability, gender-differences, gender-rate, is-baby, evolves-from, hatch-count, description.

A brief description of the data, pokemon are separated via pokedex numbers, however, pokemons’ alternate forms and mega evolutions do not have pokedex numbers of their own. The Veekun data set rectified this by assigning those without numbers numbers over 1000. Also, some pokemon only have 1 type, or less than 3 abilities. In these situations I left the attributes empty. In the original Veekun file, they avoided using null values because these were set in a relational database with a bridging table allowing there to be any amount of types or abilities (who knows, maybe someday there might be pokemon with 3 types). Putting this data into a text file takes out the possiblity of this, which is why I should have read it through a SQLITE C++ package.

In order to use this dataset, I created two classes: Pokemon and PC. This project is set up very similarly to Project 3, but it’s actually something interesting (kidding). PC has a method that reads in the file line by line, seperating each line by commas (my separating value) and inputting the data in using setters. Initially I tried to create a constructor that took in each of these values, however I found it easier to do it this way. One issue I ran into is checking values before trying to call the stoi() function on them to extract numbers. In order to check if the values contain numbers I created a separate function. PC uses a map in order to store all the pokemon (hence the name PC) and sort them. The sort function is not done very well, I couldn’t come up with a better solution so I settled for just inserting each pokemon into a new map with a different sort function in it. I created 7 sort functions in order to sort Pokemon by each of the base-stats or all of the base-stats. The PC can lookup Pokemon by names, ids, stats, and types. In addition to the lookups there is also a deposit function that accepts an id and a Pokemon and three different print functions, one that prints all pokemon, one that prints pokemon by the id stored in the map, and one that prints the first few pokemon. The Pokemon class contains all the real information on the pokemon, everything that the database contains. It also contains two check statements and several print statements. The check statements are there in order to be used by PC to generate the lists of Pokemon by stats.

If I had the time to create a GUI interface for this application, I would have windows for searching pokemon by types, searching pokemon by names and ids, searching for pokemon by stats, and looking at all pokemon. The window for searching by stats could contain sliders, and I could create a function that searches for pokemon with stats less than the input values. I could even utilize the icon images and full images located on Veekun, and make is so that when pokemon are looked up it works in a similar fashont o the application I created in R. Pulls up the icon images, and once the icons are clicked the rest of the information pops up in the window.