Ariel Espinoza

CIS 287, A. Calles

Oct 10, 2023

Midterm Report

Protein Synthesis Simulator

The field of biology, especially within informatics and data science is a difficult area for undergraduate students to get involved in. The knowledge floor needed in order to complete and understand basic tasks is typically high; an understanding of biological, chemical, and computational fields is necessary to participate in any meaningful work. For these reasons, along with various accounts of anecdotal evidence,, for many individuals, it can be exceptionally difficult in order to begin working with bioinformatic platforms. Not only are they extremely new-user unfriendly, but their user interfaces can be extremely confusing even for seasoned individuals. Take either “ClinVar” (a database for malignant human genome mutations) or a “Genome Browser” for example. Both can be exceptionally difficult to get used to especially if one is unfamiliar with amino acids as most mutations are written in the form “p.[amino acid 1][position of mutation][mutation]”. An example being “p.Ala21Gly” signifying a protein mutation from alanine to glycine on the 21st amino acid in the sequence. This is the basis for my program. For this midterm project, I chose to do a protein synthesis simulator. With this program, users will be able to practice and learn new skills regarding protein synthesis along with understanding basic biological concepts, such as amino acid properties and building basic proteins. Additionally, students will be able to use the program to get a feel for what occurs in protein synthesis for a “wet-lab” experience, being able to change -C ends of the amino acid (protein sequences are ordered from C being the first amino acid to N being the last) for any protein, completely take out particular amino acids (simulating a microRNA using RNA induced splicing complexes to remove targetted amino acids in proteins),

and more. With basic usage, the user will be able to get familiar with concepts and amino acid properties to be able to help them transition to more established interfaces.

The program satisfies all project requirements stated in the instructions. The usage of conditionals and loops can be seen in various instances throughout the code of the project. Conditional statements can be easily found within both the main file of the program and the “AminoAcid.cpp” file. Else-if statements were used to create the menu interface of the program found in main.cpp. Else-if statements were used as I added menu options to the main file as I continued with the code, iterating on it each time a new feature as worked on. Within AminoAcid.cpp, switch statements were used in place of else-if statements in order to create an option for identifying amino acids based on their one-letter code. I opted for a switch statement as there were going to be various options from the beginning (20 amino acids, meaning 20 different case options). Loops can be found all throughout the program. The first and most obvious case would be the while loop that contains most if not all of the main function code for the user interface. A while loop is set to true in order for the program to run, switching to false once the user prompts the interface to exit the program, which causes the program to stop running on a loop and close.

Classes and functions are used all throughout the program specifically in the AminoAcids and Proteins classes. The amino acid class is used to create amino acid objects containing basic information about amino acids, such as their name, charge, weight, and letter code. The Protein class is implemented to take an array of “AminoAcids” created by the user and create a vector array of them. The protein class has different functions than the amino acid class as different operations are used for proteins than amino acids in real-world situations. Functions range from simple setter and getter functions to more complex tools such as a function to remove all amino acids of the same kind.

The implementation of arrays and vectors can be quickly seen in the Protein class, as the class is made up of a vector of amino acid objects. Additionally, within the

main file of the function, a vector of Protein objects is made in order to have an organized way to store information about different proteins that the user is working on.

I/O Streams can be seen all throughout the code of the program in instances such as “cout” and “cin”. Similarly, file streams can be observed within the program, specifically within the main file, at the beginning and end of the main code. In the beginning, the file stream opens a text file and reads the information in it in order to create amino acid and protein objects. At the end, the file is edited with the current amino acids and proteins that the user has created throughout their usage of the program.

Operator overloading has been added to the protein class. In order to check if a protein is greater than, less than, or equal to another protein, the program will check how long the vector array of amino acids is within the two proteins being compared then return a boolean based on the conclusion.

The main challenges faced throughout the program involve functions that work with the vectors with the program. As the vector arrays work a bit differently than static objects for a program, many times a function did not do what was intended due to how the vectors were ordered. In order to solve this, for many instances I had to write out print statements for many of my functions in order for it to print out what it was actually doing each step of the way, especially for loops, in order to get an understanding of why certain things were not working as they were intended. An example of this was my menu option which shows the data for each protein. The loop would simply spit out just the name for the first object then return the rest of the items as undefined. Eventually, I was able to fix this issue with the help of the print statements and much googling. Another big challenge faced throughout the project was getting the whole program to run. As we are used to having one-file programs, issues with the compatibility of the files were not uncommon during development.

Once fully developed and released the program will be able to help students, and those interested, to begin using biological informatic tools used in real-life research settings. From experience from classes I have taken before, it would have been extremely beneficial to have a system that gets one used to the style and syntax of proteins. Additionally, those interested could use the program to store and write down proteins they have made or have an interest in, having functions to tell them important metrics of each protein (such as total length, charge, and weight), for projects, labs, and work.

In conclusion, the development of this program has been an exceptionally beneficial experience. Being able to work on a big project like this one has taught me various things along the way, especially when running into bugs and unexpected errors. It seems that for program development, much of what is taught in the classroom is greatly solidified during work, as one has to understand what is going on with the code on a decent level in order to make it do what you originally intended it to.