In this project, John Leone and Ahsan Virk have worked together to find the genetic algorithm to a knapsack problem. The way the work was divides was mostly in two parts. A major part of documenting what we do was the responsibility of John Leone, while Ahsan Virk took care of how we would implement our code. Throughout the assignment, John Leone helped Ahsan Virk in programming the code, suggesting ideas which lead to fewer errors. Ahsan Virk on the other side, helped John Leone in finishing up the documentation when he was done with programming.

**Introduction (John & Ahsan)**:

In this project, Ahsan Virk and John Leone worked together to form a genetic algorithm that would solve the knapsack problem. We wanted to see what the best combination of items would be if we had a 30-pound weight limit for what we could pack. The best solution would consist of the maximum amount of survival points, helping our client’s survival since that particular combination of items would carry the most power in their defense when compared to all the other items weighing under 30 pounds.

**Procedure (John)**:

The four extra items we decided to add were a pistol crossbow, compass, solar powered radio, and a backpack. Since more survival points meant a higher chance of survival, the solution with the highest amount of survival points would be our best solution. Knowing that, we decided that any item whose survival points divided by their weight was less than 1 would be barred from being in any solution. As a result, solutions with bottle, sugar candy or both were eliminated. The total weight of every solution we came up with was 30 pounds. We decided to add each item’s respective binary number to make implementing our algorithm easier.

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Binary Number** | **Weight** | **Survival Points** |
| **Sleeping Bag** | **0001** | **15** | **15** |
| **Rope** | **0010** | **3** | **7** |
| **Pocket Knife** | **0011** | **2** | **10** |
| **Flashlight** | **0100** | **5** | **5** |
| **Bottle** | **0101** | **9** | **8** |
| **Sugar Candy** | **0110** | **20** | **17** |
| **Pistol Crossbow** | **0111** | **8** | **13** |
| **Compass** | **1000** | **2** | **9** |
| **Solar Powered Radio** | **1001** | **12** | **15** |
| **Backpack** | **1010** | **9** | **14** |

**Discussion (Ahsan)**:

There wasn’t any output we came across while doing this project because there was a runtime error we didn’t know how to fix. However, from what we know about genetic algorithms, we were expecting to see different solutions every time we ran the code, since it was meant to be programmed in a way which moves on to the next best solution.

**Conclusion (Ahsan & John)**

Some things that went smoothly during the project included assigning weights and survival points to the items. After that, we decided to go line by line of the assignment on canvas, and as we did that, we made notes on what we need in our program. That really helped us in getting started with the implementation of the algorithm. However, we faced severe blocks while writing this piece of code and had to do some research on how a genetic algorithm is implemented correctly, since programming an artificial intelligence problem is still new to us. We had trouble finding runtime errors that caused our entire program to not work. Not to mention the syntax errors, which were quicly resolved and weren’t as much of a problem when compared to figuring out the run-time errors which aren’t visible on the program.