**How we designed system features:**

For the majority of the project, our design philosophy followed:

1. Analysis of the task at hand
2. Applying knowledge learnt from lectures and other resources, such as, stackoverflow, etc
3. Implementing the code
4. Rigorous testing with a multitude of cases to find bugs within our code
5. Amending functions and related code to 1. Remove bugs 2. Make code more efficient and robust, 3. Something

For example, for our menu, we initially had an if statement checking the validity of each possible error case, and adding to the condition with every error we found. However, this grew very large and we had an idea to create a function that would sort and remove every non integer in an input, which we could then use a simple if statement to ensure that the input was valid corresponding to the menu selection. This allowed us to ensure inputs were valid and made our code much more efficient, as we no longer had to paste a large if statement each time we wanted an input.

**User interface:**

Menu system implemented with a while loop. Based on our selector, the menu shifts to various functions. We use >> as our cursor of choice. In addition, we also make use of ---- and line breaks to create space and separate function and menu to make the interface clearer and easier to read

This is our menu. It opens this on startup and appears after a function is done processing and an output has been displayed. We also make use of submenus within each option to keep our menu simple and clean.

**Core classes and functions:**

Our essential classes include

1. Our information classes, known as Domestic, International, and Student. These include all information relating to students, split by classification. This includes research score, area of origin, and more.
2. Our classes that contain functions to create linked lists , this includes RS, area of origin and more

Our essential functions include

1. Stu\_sort, our merge sort function, initially used quick sort, but once we swapped to linked lists, we found that merge sort was more efficient.
2. Get\_number
3. Findfunctions that make the list

**Our innovation:**

In order to prevent re-structuring and writing the files, since we originally used arrays we decided to convert the arrays to linked lists and then delete the arrays in order to maintain the linked lists. Moreover, we separated them based on classes.

* International students must have a minimum TOEFL score of 93 with at least a 20 in each category, all those that don’t pass this are deleted.

Can show rejected international students, sort them according to first and last name, cgpa, research score, and country. We do this by storing the rejected students into a list when we initially filter the international students

Can show merged students, and sort them according to first and last name, cgpa, research score, and type.

Another innovation of ours is when all students are printed, it classifies them by domestic / international status

Threshold for all lists  
showcase head and tail nodes

**Project Management:**

We worked on everything together, made use of online resources including discord and vscode liveshare. We used a private github repo to organize all our files and version control

**Lessons learned from the project:**

Time management skills, professional use of github which will benefit us for co-ops and future classes,