Group Responsibilities:

Nicholas: compare function, overload << function, use arrays/vectors
Cole: second sorting algorithm
Mika: testing, error checking, main menu, writing pdf on who did what
Patrina: first sorting algorithm
Jessica: class creation

Weekly Progress:
Start Date: Oct 23

Oct 23: Divided up tasks appropriatley
Oct 24: Started with setting up the classes as well as implementing the basic class functions (get, set, and constructo rs) and started out lining how to do the sorting functions
Oct 25: Implemented the remaining class (compare) functions and started on setting up main properly for the user in put and to be able to use the sorting functions properly
Oct 26: Implemented the sorting functions for all the attributes but noticed a few errors
Oct 27: Implemented the a large amount of the user input portion but still had error checking to do
Oct 28: Finalized the user input section and wrinkled out the errors with the sorting. Added various levels of error c hecking into the user input.

Testing and test cases:

For testing we tried various different cases and from the results we accommodated for various different inputs

Regular cases:
- Legal numeric inputs
- Legal numeric inputs that have no function within the user input section
- Typical sorting operations. eg. sorting the default vector for the first time

Corner cases:
- Sorting the same array mulitple times, before and after using the sort all function. Especially for the international students vector for which the sort all returns a vector that removes certain students
- Repeatedly switching between which student vector to operate on

Illegal cases:
- Illegal non-numeric inputs. eg, chars when the expected input is an integer
- Illegal inputs such as blank lines and spaces
- Valid inputs concatenated with illegal inputs. eg '1g' when the input should just be a number such as '1'

The results of these cases allowed us to create a good user input that can recognize and properly behave on various different types of inputs.