CS 31 Project 4 Report

**Notable obstacles faced**:

1. Figuring out how the specific functions should behave when given an array with no elements was a significant obstacle I had to overcome.
2. Making sure to prevent undefined behavior was another obstacle I faced.
3. Understanding exactly what each of the functions were required to do and return as per the project spec was a challenge as well. I had to ensure that all cases were dealt with.

**Test Cases:**

string people[5]={"clark","peter","diana","tony","selina"};

1. appendToAll(people,5,"!!!") **// Check if it appends to all elements**
2. appendToAll(people,3,"???") **//Check if it appends to specified number of elements**
3. appendToAll(people,0,"...") **//Check behavior with 0 interesting elements**
4. appendToAll(people,-3,"///") **//Check behavior with negative number of elements**

string names[6]={"Noah","Rishab","Athu","Pali","Rishab","Rishab"};

1. lookup(names,-3,"Ammar") **//Check behavior with negative number of elements**
2. lookup(names,0,"Ammar") **//Check behavior with 0 interesting elements**
3. lookup(names,3,"Pali") **//Check behavior with specified number of elements**
4. lookup(names,6,"Rishab") **//Check behavior with string that appears multiple times**
5. lookup(names,6,"AThU”) **//Check behavior with incorrect case**
6. positionOfMax(names,6) **//Find greatest string from all elements when largest repeats**
7. positionOfMax(names,0) **//Check behavior with 0 interesting elements**
8. positionOfMax(names,-3) **//Check behavior with negative number of elements**
9. rotateLeft(names,6,3) **//Check behavior with all elements**
10. rotateLeft(names,4,2) **//Check behavior with specified number of elements**
11. rotateLeft(names,3,4) **//Check behavior when pos exceeds number of interesting elements**
12. rotateLeft(names,0,0) **//Check behavior with 0 interesting elements**
13. rotateLeft(names,-3,0) **//Check behavior with negative number of elements**

string names2[7]={"Rohan","Rohan","Rohan","Rishan","Zaid","Akshay","Akshay"};

1. countRuns(names2,4) **//Check behavior for specified number of elements**
2. countRuns(names2, 7) **//Check behavior for all elements**
3. countRuns(names2,0) **//Check behavior with 0 interesting elements**
4. countRuns(names2,-3) **//Check behavior with negative number of elements**
5. flip(names2,7) **//Check behavior with all elements**
6. flip(names2,3) **//Check behavior with specified number of elements**
7. flip(names2,0) **//Check behavior with 0 interesting elements**
8. flip(names2,-3) **//Check behavior with negative number of elements**

string folks[6]={"bruce", "steve", "", "tony", "sue", "clark"};

string group[5]={"bruce", "steve", "clark", "", "tony"};

1. differ(folks,6,group,5) **//Check behavior with all elements**
2. differ(folks,4,group,2) **//Check behavior with specified number of elements**
3. differ(folks,-3,group,2) **//Check behavior with negative number of elements**
4. differ(folks,0,group,2) **//Check behavior for 0 interesting elements**

string names3[10]={"logan","reed","sue","selina","bruce","peter"};

string names4[10]={"reed","sue","selina"};

1. subsequence(names3, 6, names4, 3) **//Check behavior for subsequence present**

string names5[10]={"logan","selina"};

1. subsequence(names3, 6, names5, 2) **//Check behavior for no subsequence present**

string names6[0];

1. subsequence(names3,6,names6,0) **//Empty string array is a subsequence of any string array**
2. subsequence(names3,6,names4,0) **//Empty string array is a subsequence of any string array**

string names7[3]={"bruce","peter","sue"};

1. subsequence(names3,6,names7,3) **//Check behavior with all elements**
2. lookupAny(names3, 6, names7, 3) **//Check behavior with all elements**
3. lookupAny(names3,6,names6,0) **//Check behavior with no interesting elements**
4. lookupAny(names3, 6, names, 4) **//Check behavior with specified number of elements**
5. lookupAny(names5, 2, names7, 3) **//Check behavior with specified number of elements**
6. split(names3,4,"quirky") **//Check behavior with specified number of elements**
7. split(names3,6,"quirky") **//Check behavior with specified number of elements**
8. split(names3,6,"zeal") **// Check behavior with no elements >= splitter**
9. split(names3,-3,"zeal") **//Check behavior with negative number of elements**
10. split(names3,0,"zeal") **//Check behavior with 0 elements**

string names8[5]={"rock","paper","scissors","rock","rock"};

1. split(names8,5,"rock") **//Check behavior with multiple elements == splitter**