1. Some notable obstacles:

* Shifting an array backwards in order to “delete” an element of the array
* How to find the subsequence of one array by simultaneously looping through two arrays
* How to divide and sort an array given an divider (sorting them from left and right)

1. Test Cases:

**Enumerate:**

string fruits[8] = {"watermelon", "apple", "cherry", "banana", "blueberry", "apple", "guava", "pear"};

int test = enumerate(fruits, 8, "watermelon");

cout<<test;

**Locate:**

string fruits[8] = {"watermelon", "apple", "cherry", "banana", "blueberry", "apple", "guava", "pear"};

int test = locate(fruits, 8, "watermelon");

cout<<test;

string fruits[8] = {"watermelon", "apple", "cherry", "banana", "blueberry", "apple", "guava", "pear"};

int test = locate(fruits, 8, "durian");

cout<<test;

**Locate Sequence:**

string fruits[8] = {"watermelon", "apple", "banana", "banana", "blueberry", "apple", "guava", "pear"};

int b;

int c;

bool tester;

tester = locateSequence(fruits, 8, "banana", b, c);

cout<<tester<<endl;

cout<<b<<endl;

cout<<c<<endl;

string fruits[5] = {"apple", "melon", "pear", "strawberry", "apple"};

int p;

int q;

bool fruity;

fruity = locateSequence(fruits, 5, "apple", p, q);

cout<<fruity<<endl;

cout<<p<<endl;

cout<<q<<endl;

string fruits1[7] = {"apple", "apple", "banana" , "booty", "melon", "pear" , "moon"};

bool testa;

int test;

int test1;

testa = locateSequence(fruits1, 7, "banana", test, test1);

cout<<testa<<endl;

cout<<test<<endl;

cout<<test1<<endl;

**LocationOfMin:**

string fruits[5] = {"apple", "apple", "pear", "strawberry", "durian"};

int p;

p = locationOfMin(fruits, 5);

cout<<p;

**MoveToEnd:**

**Using the function:**

void showStrings(const string s[], int n) {

cout << "string[" << n << "] = [";

for (int i = 0; i < n; i++) {

cout << s[i] << ",";

}

cout << "]"<< endl;

}

(Prints out the new array)

string fruits[5] = {"apple", "apple", "pear", "strawberry", "durian"};

int p;

p = moveToEnd(fruits, 5, 0);

cout<<p;

showStrings(fruits, 5);

**MoveToBeginning:**

string fruits[5] = {"apple", "apple", "pear", "strawberry", "durian"};

int p;

p = moveToBeginning(fruits, 5, 4);

cout<<p;

showStrings(fruits, 5);

**LocateDifference:**

string fruits[5] = {"apple", "apple", "pear", "strawberry", "durian"};

string fruits1[7] = {"apple", "banana", "pear" , "donut", "durian", "mango" , "guava"};

int p;

p = locateDifference(fruits, 5, fruits1, 7);

cout<<p;

string fruits[5] = {"apple", "apple", "pear", "strawberry", "durian"};

string fruits1[7] = {"apple", "apple", "pear" , "strawberry", "durian", "mango" , "guava"};

int p;

p = locateDifference(fruits, 5, fruits1, 7);

cout<<p;

**EliminateDups:**

string fruits1[7] = {"apple", "apple", "pear" , "pear", "pear", "mango" , "mango"};

int p;

p = eliminateDups(fruits1, 7);

cout<<p;

showStrings(fruits1, 7);

**Subsequence:**

string fruits[3] = {"apple", "apple", "pear"};

string fruits1[7] = {"apple", "apple", "banana" , "guava", "melon", "pear" , "mango"};

bool test;

test = subsequence(fruits1, 7, fruits, 3);

cout<<test;

string fruits[3] = {"banana", "apple", "pear"};

string fruits1[7] = {"apple", "apple", "banana" , "guava", "melon", "pear" , "mango"};

bool test;

test = subsequence(fruits1, 7, fruits, 3);

cout<<test;

**MakeMerger:**

string fruits[3] = {"banana", "apple", "pear"};

string fruits1[7] = {"apple", "apple", "banana" , "guava", "melon", "pear" , "mango"};

string result[10];

int test;

test = makeMerger(fruits1, 7, fruits, 3, result, 10);

cout<<test;

**Doesn’t work since arrays are not nondecreasing.**

string fruits[3] = {"apple", "apple", "banana"};

string fruits1[8] = {"apple", "apple", "banana" , "guava", "melon", "pear" , "strawberry", "strawberry"};

string result[10];

int test;

test = makeMerger(fruits1, 8, fruits, 3, result, 10);

cout<<test;

showStrings(result, 10);

**Doesn’t work since the size of the two arrays are larger than**

**the final array.**

string fruits[3] = {"apple", "apple", "banana"};

string fruits1[7] = {"apple", "apple", "banana" , "guava", "melon", "pear" , "strawberry"};

string result[10];

int test;

test = makeMerger(fruits1, 7, fruits, 3, result, 10);

cout<<test;

showStrings(result, 10);

**Divide:**

string g[4] = { "samwell", "margaery", "tyrion", "jon" };

int s = divide(g, 4, "margaery");

cout<<s<<endl;

showStrings(g, 4);

string g[10] = { "samwell", "margaery", "tyrion", "jon", "carter" ,"alison","khaw","valerie", "khaw", " "};

int s = divide(g, 10, "margaery");

cout<<s<<endl;

showStrings(g, 10);