Partners: Sophia Shah and Sriahalya Thirumurugan

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

//assuming the input is a 2D vector
vector<string> returnUnique (vector<vector<string>> inputArr){
    vector<string> returnedVect;
    int rows = inputArr.size();
    int cols = inputArr[0].size();

    //check if input vector is blank, if not then add first element to returned vector
    if (rows > 0 && cols > 0){
        returnedVect.push_back(inputArr[0][0]);
    }
    else{
        return returnedVect;
    }

    //loop through input vector
    for(int i=0; i<rows; i++){
        for(int j=1; j<cols; j++){
            string currElement = inputArr[i][j];

            //checks if current element is already in the returned vector
            for(int k=0; k<returnedVect.size(); k++){
                if(currElement == returnedVect[k]){
                    continue;
                }
            returnedVect.push_back(currElement);
            }
        }
    }
    return returnedVect;
}
```

Partners: Sophia Shah and Sriahalya Thirumurugan

PART 2

Time Complexity:

Our worst case time complexity is O(n*m*k), where n is the number of rows in the given vector, m is the number of columns in the given vector, and k is the number of elements in the returned vector. In the worst case scenario, all elements in the given input are unique, meaning the for loop has to run the size of the returned vector (k) amount of times for each element in the 2D vector (n*m).

Space Complexity:

Our worst case space complexity is O(n*m*t) where n is the number of rows in the given vector, m is the number of columns in the given vector, and t is the maximum number of characters allowed for each element. In this worst case scenario, all elements from the given vector are unique and every single one has to be added to the returned vector. All elements are also the maximum length of characters in the worst case scenario.