CS 413 Homework-1

Question 1)

a) Define an algorithm. Explain its properties.

- An algorithm is a step-by-step procedure for solving a problem or accomplishing some end.
- It is a finite sequence of an unambiguous, executable steps that ultimately terminates if followed.

<u>Properties of Algorithms:</u>

- It should terminate after a finite time.
- It should produce at least one output.
- It should take zero or more input.
- Every step in the algorithm must do some work.

b) Consider a problem in people's daily life that has been solved by an algorithm. Explain the problem and the algorithm used to solve the problem. Do you have any better idea to solve the same problem? Explain why?

- A common daily life problem in people's daily life is to determine the best route among all which will be the most convenient and make them reach the destination in the shortest possible time.
- This problem is solved by GPS based navigation apps, which used Djikshtra's Algorithm to determine the shortest path between the two destinations based on the most recent traffic conditions and road networks.
- Before GPS based navigation apps, people relied on the static maps or personal experience to some unmoved landmarks, which often led to inefficient travel time.

<u>Djikshtra's Algorithm:</u>

- Modern navigation apps like Google Maps and Waze use Djikshtra's Algorithm to determine the shortest path to a destination. The algorithm works as follows:
  1. Assigning an initial distance of zero the starting node and infinity to all others.
  2. Exploring the nearest unvisited node and updating the shortest known distances to its neighboring nodes.
  3. Repeating this process until it reaches the destination.

<u>A better alternative?</u>

- Undoubtedly, Djikshtra's Algorithm is effective, yet it can be proved to be expensive in terms of computations for large scale networks.
- A more efficient approach will be the use of The Search Algorithms, which will improve upon Djikshtra by using heuristics such as an estimated distance to the goal to help prioritize the path.
- This will reduce the unnecessary computations making it faster and more effective.

Question 2)

a) Define an instance of a problem in general.

- A problem is a general question to be answered.
- An instance of a problem is obtained by specifying particular values for all the problem parameters.

b) Consider the following Summation problem: Given an integer list (an array) of size n, we want to calculate the summation of all numbers in the list and display the result.

(1) Specify two different instances of the Summation problem defined above.

- Instance 1: Given an integer list A = [3,7,1,9,4], find the sum of all the elements.
- Instance 2: Given an integer list B = [-2,5,8,-1,0,6], find the sum of all the elements.

(2) What is the solution for each instance in part?

- The solution to the instance 1 is 3+7+1+9+4 = 24.
- The solution to the instance 2 is -2+5+8+(-1)+0+6 = 16.

(3). How did you come up with that solution?

- The summation algorithm follows a straightforward iterative approach:

1. Initialize a variable S to 0.
2. Iterate through every element in the list of integers and add it to the S.
3. Return the final value of S.

Question 3)

a) Write an algorithm for sorting a list of integer numbers using the bubble sort algorithm

- Bubble Sort Algorithm

<u>First Iteration:</u>

1. Starting from the first index, compare the element at the current index with the element at the next index.
2. If the element at the current index is greater than the element at the next index, swap the elements.
3. Repeat this process till the index of the last element.

<u>Remaining Iterations:</u>

1. The process goes the same for the remaining iterations.
2. After each iteration, the largest element among the unsorted elements is placed at the end.
3. In each iteration, the comparison takes place up to the last sorted element.
4. The array is sorted after all the unsorted elements are placed at its correct position.

<u>Psuedo Code:</u>

```
_ BubbleSort(arr, n):
1. for i = 0 to n-1:
2.     for j = 0 to n-1-i:
3.         if arr[j] > arr[j+1]:  // Swap if the left element is greater
4.             swap(arr[j], arr[j+1])
5. return arr
```

b) Assume you are given the numbers 4, 2, 3, 1. Show step by step how your described algorithm in part (a) works on the given list of numbers and moves them through the list until the algorithm terminates and the list is sorted in increasing order.

- Initial List: [4,2,3,1]
- Pass 1:
  1. Compare 4,2. Since 4 > 2, swap 4,2.
  2. Updated list: [2,4,3,1].
  3. Compare 4,3. Since 4 > 3, swap 4,3.
  4. Updated list: [2,3,4,1].
  5. Compare 4,1. Since 4 > 1, swap 4,1.
  6. Updated list: [2,3,1,4].

- Pass 2:
  1. Current list: [2,3,1,4].
  2. Compare 2,3. Since 2 < 3, no swapping is done.
  3. Compare 3,1. Since 3 > 1, swap 3,1.
  4. Updated list: [2,1,3,4].

- Pass 3:
  1. Current list: [2,1,3,4].
  2. Compare 2,1. Since 2 > 1, swap 2,1.
  3. Updated list: [1,2,3,4].
  4. List is completely sorted.

- Final sorted list: [1,2,3,4]

c) Now analyze the time complexity of the bubble sort algorithm STEP BY STEP as what we did for the linear search. Show all your work. Then specify the worst-case time complexity of bubble sort in Big-O notation.

- Time Complexity of Bubble Sort:
  1. Count the number of comparisons:
     - On the first pass, we compare n-1 elements.
     - On the second pass, we compare n-2 elements.
     - On the third pass, we compare n-3 elements.
     - This is continued till we compare only one element.

Therefore, the total number of comparisons are:

(n-1) + (n-2) + (n-3) + (n-4) + ...+ 1 = (n*(n-1)) /2.

Using Big O notation, we approximate it to ***O(n^2)***.

2. Count the number of swaps:
    - Counting the number of swaps follows the same pattern as counting the number of comparisons.
    - When the list is in descending order, every pair must be swapped.
    - Thus, the total number of swaps is approximated to ***O(n^2).***

- Worst Case time Complexity:
    - The dominant term in the sum is n^2.
    - Hence, the worst case time complexity of the Bubble Sort is ***O(n^2)***.

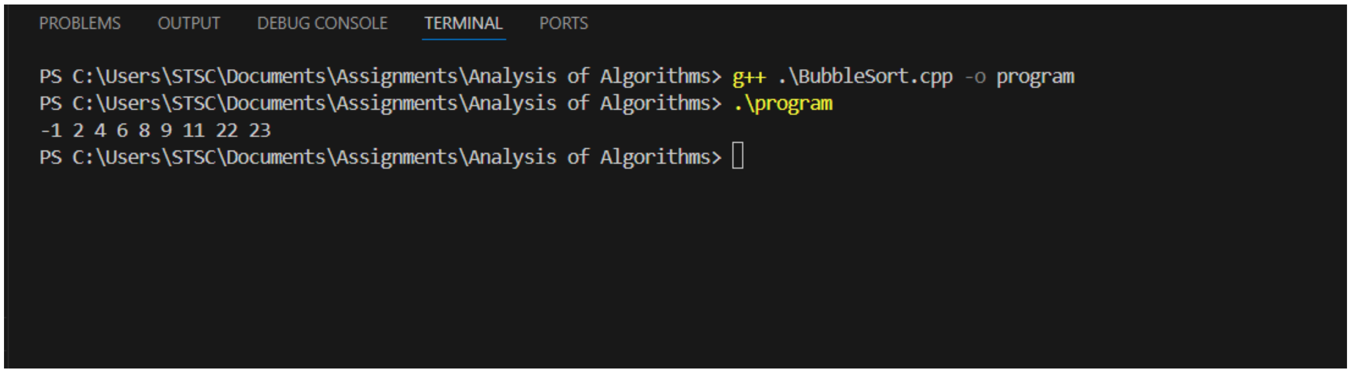
Question 4)

Write a complete program in C++/Java for the bubble sort algorithm in Question 3. Paste your complete code here as the solution. Also, share 3 screenshots of the output you get from your program once you run your code for the following inputs:

Input #1: 11, 23, 2, 4, 6, 22, 8, 9, -1
(we expect these numbers to appear in ascending order as the output of your code, i.e., -1, 2, 4,6, 8, 9, 11, 22, 23
Input #2: 15, -1, 56, 34, 22, 8, 48, 1
Input #3: 6, -9, -15, 12, 38, 4, 11

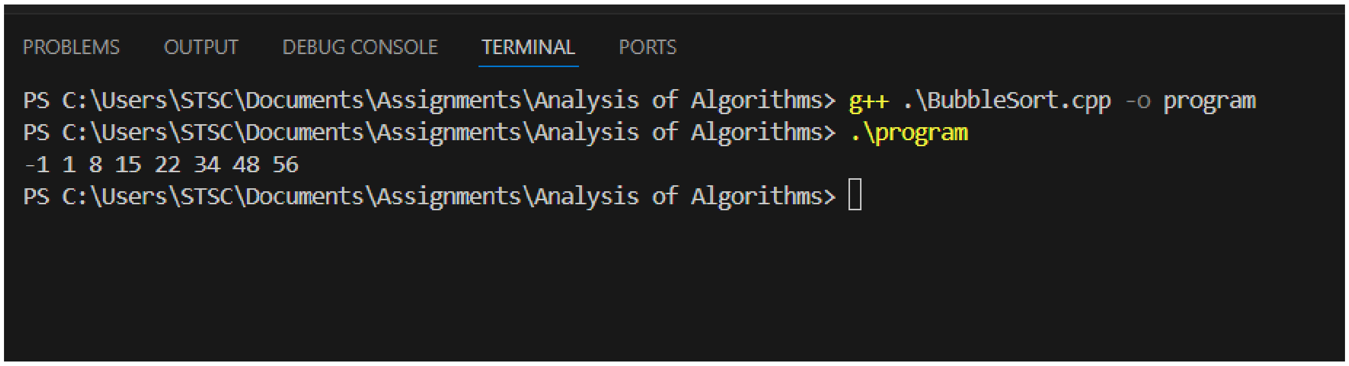
- Program for Bubble Sort in C++:

```
#include<iostream>
#include<vector>

using namespace std;

void bubbleSort(vector<int>& arr){
    for(int i = 0; i<arr.size();i++){
        for(int j = 0; j<arr.size()-i-1;j++){
            if(arr[j] > arr[j+1]) swap(arr[j],arr[j+1]);
        }
    }
}

int main(){
    vector<int>input_1 = {11, 23, 2, 4, 6, 22, 8, 9, -1};
    vector<int>input_2 = { 15, -1, 56, 34, 22, 8, 48, 1};
    vector<int>input_3 = {6, -9, -15, 12, 38, 4, 11};
    bubbleSort(input_3);

    for(int a: input_3){
        cout<<a<<" ";
    }
    return 0;
}
```



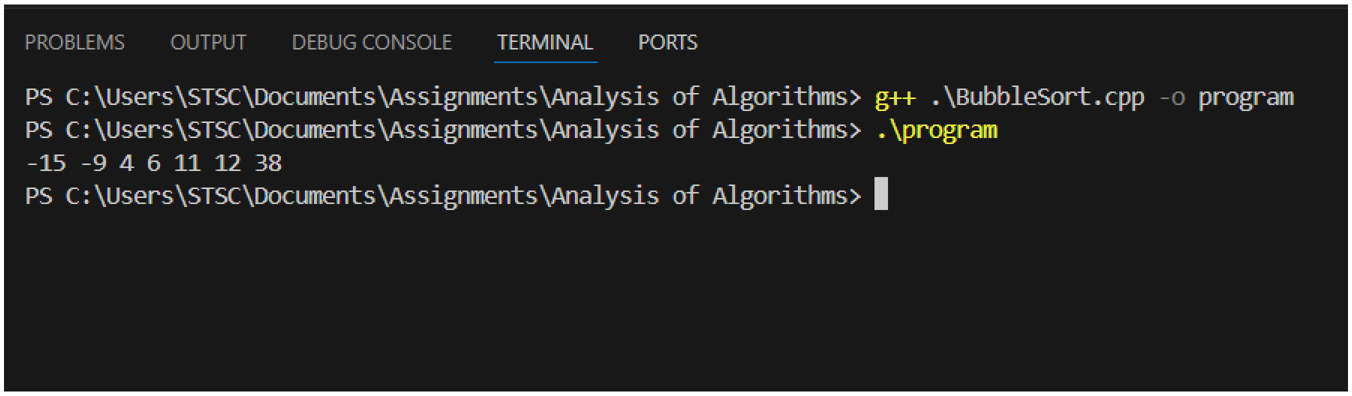
```
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> g++ .\BubbleSort.cpp -o program
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> .\program
-1 2 4 6 8 9 11 22 23
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> 
```

Figure 1: Output for list = {11, 23, 2, 4, 6, 22, 8, 9, -1}



```
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> g++ .\BubbleSort.cpp -o program
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> .\program
-1 1 8 15 22 34 48 56
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> 
```

Figure 2: Output for list = {15, -1, 56, 34, 22, 8, 48, 1}



```
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> g++ .\BubbleSort.cpp -o program
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> .\program
-15 -9 4 6 11 12 38
PS C:\Users\STSC\Documents\Assignments\Analysis of Algorithms> 
```

Figure 3: Output for list = {6, -9, -15, 12, 38, 4, 11}