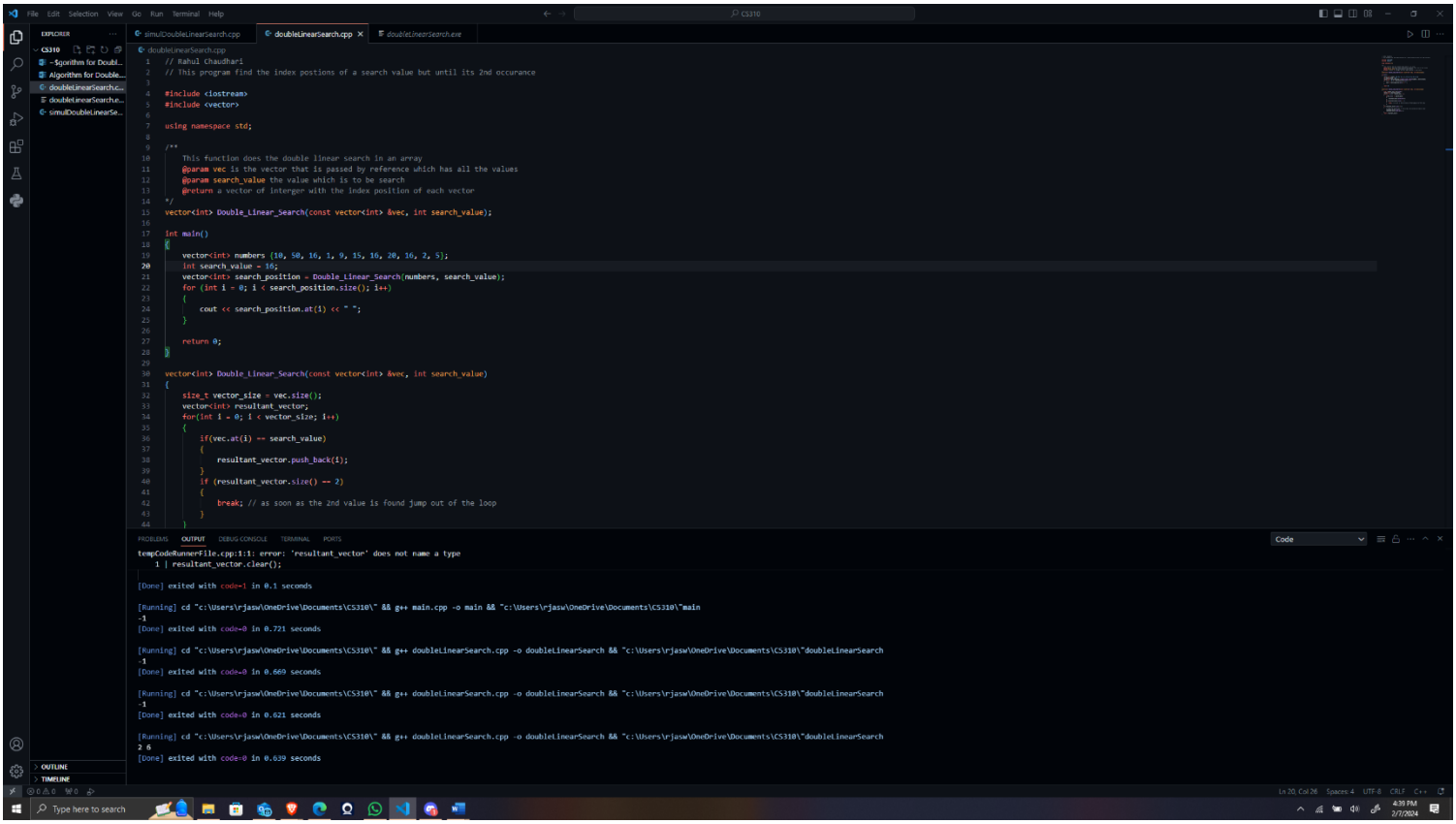
Deliverable 1: pseudo code algorithm for double linear search

Return type Vector of integers, Function name (vector with const and reference, search_value)

```
 Size_t vector_size := size of vector
Initialize an empty vector resultant_vector
for i from 0 to  less than vector_size  do
    if vec[i] is equal to search_value then
        Append i to resultant_vector
    end if
    if size of resultant_vector is equal to 2 then
        break
    end if
end for
if size of resultant_vector is less than 2 then
    Clear resultant_vector
    Append -1 to resultant_vector
end if
return resultant_vector
```

Deliverable 2: Attachment of Screenshot of doublelinearSearch.cpp



Deliverable 3: Calculating the Big O for the Algorithm.

Code	Cost	Times
`vector<int> Double_Linear_Search(const vector<int> &vec, int search_value)`		
`{`		
`size_t vector_size = vec.size();`	C1, C2	1 1
`vector<int> resultant_vector;`	C3	1
`for(int i = 0; i < vector_size; i++)`	C4 (i = 0), C5(i<vec..)	1 n + 1
`{`		
`if(vec.at(i) == search_value)`	C6, (.at) C7	n n
`{`		
`resultant_vector.push_back(i);`	C8	n
`}`		
`if (resultant_vector.size() == 2)`	C9, (.size) C10	n n
`{`		
`break;`	C11	1
`}`		
`}`	C12 (i++)	n
`if (resultant_vector.size() < 2)`	C13,(.size) C14	1 1
`{`		
`resultant_vector.clear();`	C15	1
`resultant_vector.push_back(-1);`	C16	1
`}`		
`return resultant_vector;`	C17	1
`}`		

Cost Function:
C1+C2+C3+C4+C5+C6+C7+C8+C9+C10+C11+C12+C13+C14+C15+C16+C17
T(n) = 1+1+1+1+n+1+n+n+n+n+n+1+n+1+1+1+1+1
T(n) = 7n + 11

Here, our T(n) = 7n+11 and has order of O(n) because

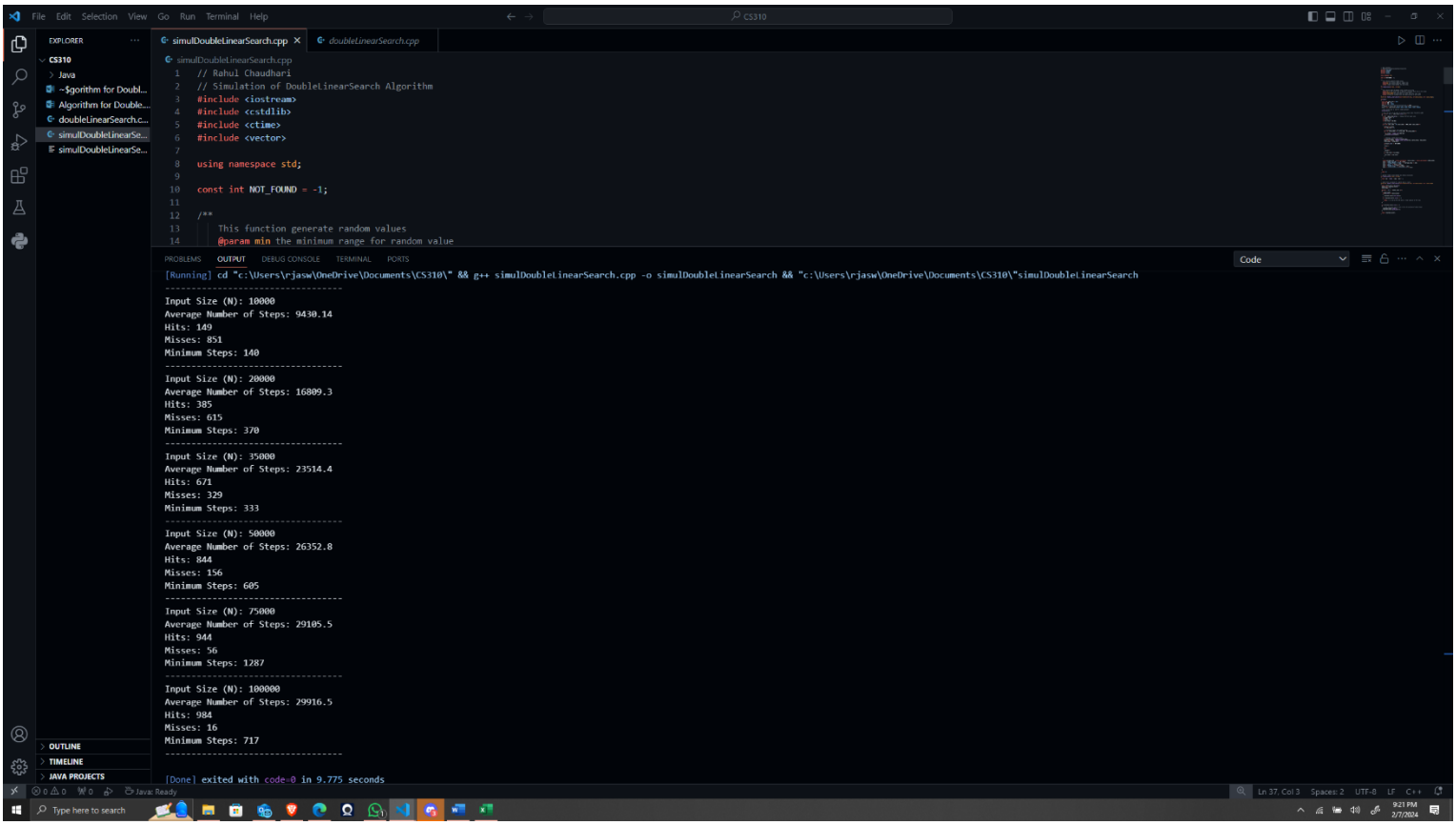
7n+11 <= 10 * n where n > n_0

For any values of n >= 10, c = 10, n_0 = 1

Here, f(n) = n

Hence the Big O would be O(f(n)) = O(n).

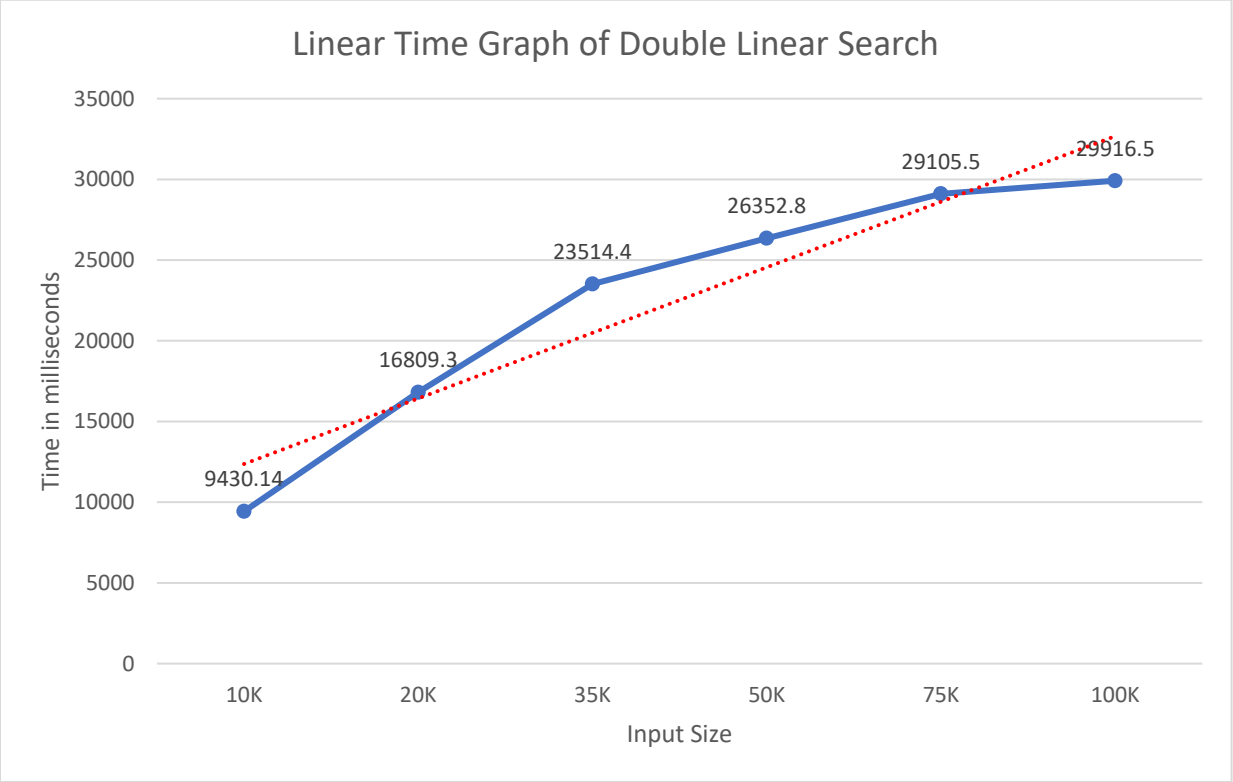
Deliverable 4: Screenshot of the Code with Simulation Data

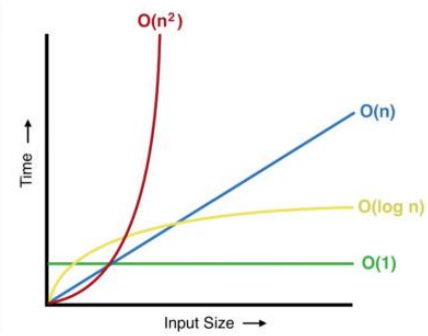


This Table represents the hits, misses, min steps and average steps for the double linear search algorithm for different input sizes ranging from 10K to 100K.
We can also notice the average steps increasing as the input sizes is increasing.

Input Size	Hits	Misses	Min Steps	Average Steps
10K	149	851	140	9430.14
20K	385	615	370	16809.3
35K	671	329	333	23514.4
50K	844	156	605	26352.8
75K	944	56	1287	29105.5
100K	984	16	717	29916.5

Deliverable 5: Graph of the Data





In this graph we can notice that the red trendline is going straight, which means the overall function is following a linear fashion. If we compare it with the Big O graph of different functions, we can see that it clearly resembles O(n), which we also got from the cost function calculation. Thus, this graph comparison, the cost function and the simulation verifies that the algorithm works in linear time.