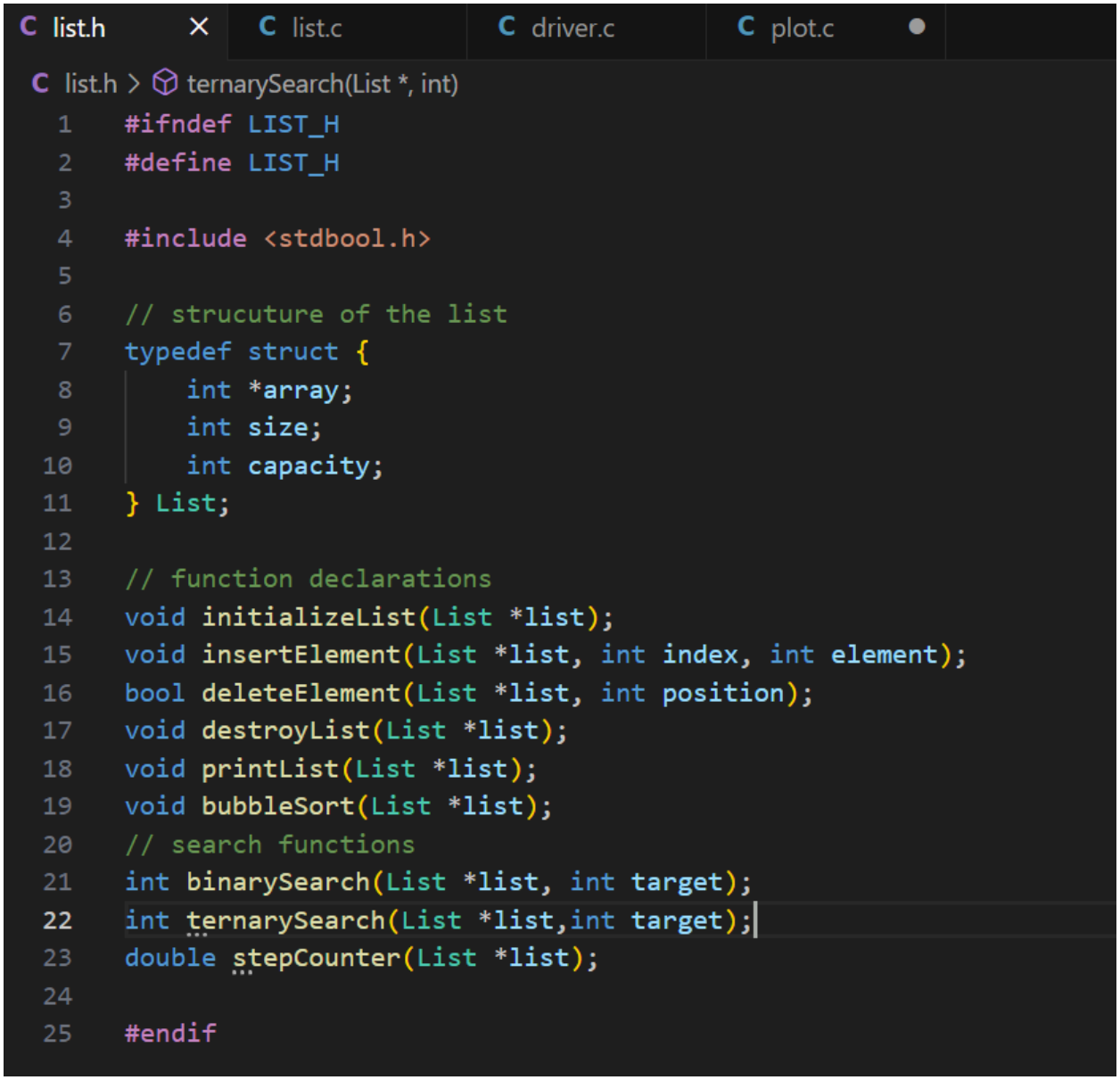
DSA ASSIGNMENT 1

Header File



List.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "list.h"

void initializeList(List *list){
    int MAX;
    printf("Enter capacity of list: \n");
    scanf("%d",&MAX);
    list->capacity=MAX;
    printf("Enter size of list: \n");
    scanf("%d",&list->size);
    list->array = (int *)malloc(list->capacity*sizeof(int));
    for (int i=0;i<list->size;i++){
        printf("Enter element no. %d: \n",i+1);
        scanf("%d",&list->array[i]);
    }
}

void insertElement(List *list, int index, int element) {
    if (index < 0 || index > list->size) {
        printf("Index %d is out of bounds. Valid index is between 0 and %d.\n", index, list->size);
        return;
    }

    // if capacity is already full
    if (list->size == list->capacity) {
        int newCapacity = list->capacity * 2;
        int *newArray = (int *)malloc(newCapacity * sizeof(int));
        for (int i = 0; i < list->size; i++) { // creating new copy of array
            newArray[i] = list->array[i];
```

```
            newArray[i] = list->array[i];
        }
        free(list->array);
        list->array = newArray;
        list->capacity = newCapacity;
    }

    // shifting
    for (int i = list->size; i > index; i--) {
        list->array[i] = list->array[i - 1];
    }
    //inserting new element and updating new size of the list
    list->array[index] = element;
    list->size++;
}
bool deleteElement(List *list, int position) {
    if (position < 0 || position >= list->size) {
        return false;
    }

    for (int i = position; i < list->size - 1; ++i) {
        list->array[i] = list->array[i + 1];
    }

    list->size--;
    return true;
}
void bubbleSort(List *list){
    for(int i=0;i<list->size-1;i++){
```

```
    for(int i=0;i<list->size-1;i++){
        for(int j=0;j<list->size-i-1;j++){
            if(list->array[j]>list->array[j+1]){
                int temp;
                temp=list->array[j+1];
                list->array[j+1]=list->array[j];
                list->array[j]=temp;
            }
        }
    }
}
int binarySearch(List *list, int target){
    bubbleSort(list);
    int low=0;
    int high= list->size-1;
    while(low<=high){
        int mid = (low+high)/2;
        if(list->array[mid]==target){return mid;}
        if (list->array[mid]>target){high = mid - 1;}
        else{low = mid + 1;}

    }
    return -1;

    }
    void destroyList(List *list) {
        free(list->array);
        list->array=NULL;
        list->capacity=0;
```

```
86          list->capacity=0;
87          printf("List is destrotyed.\n");
88      }
89
90      void printList(List *list) {
91          for (int i = 0; i < list->size; ++i) {
92              printf("%d ", list->array[i]);
93          }
94          printf("\n");
95      }
96      double stepCount(int size){
97          return log2(size+1);
98      }
99
```

Driver.c

```
#include "list.c"

int main() {
    List list;
    initializeList(&list);  //Initialize list
    insertElement(&list,3,4); //Insert element.
    printList(&list);     //Print list
    deleteElement(&list, 3);   //Delete element.
    printList(&list);
    printf("Element found at index: %d",binarySearch(&list,5));  //Binary Search
    printf("Element found at index: %d",ternarySearch(&list,5)); //Ternary Search
    destroyList(&list); // Destroying List
    printList(&list);
    return 0;
}
```

Plot

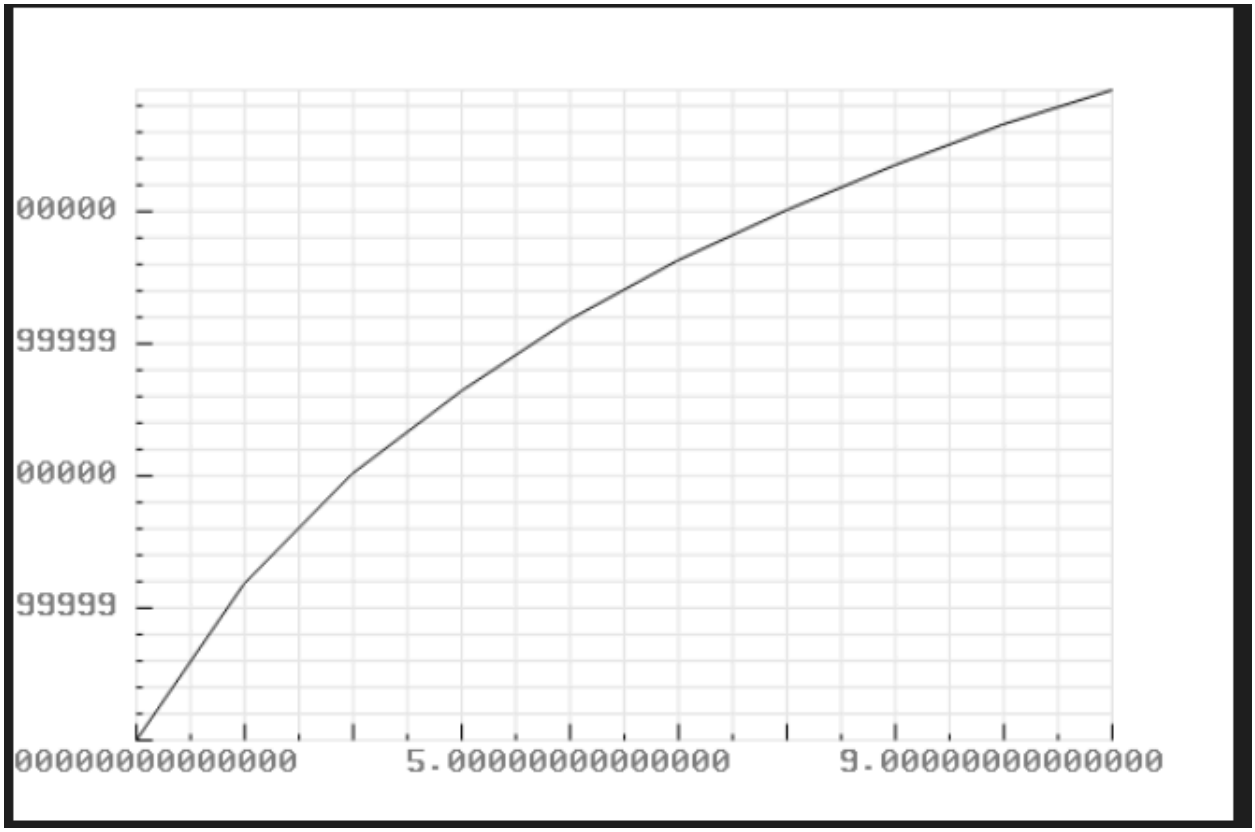
plot.c > main()

C:\Users\aarya\Desktop\dsalab\plot.c

```c
#include "pbPlots.h"
#include "supportLib.h"

int main(){
    int n;
    printf("How many size you want to input: ");
    scanf("%d",&n);
    double x[n];
    double y[n];
    for(int i=0;i<n;i++){
        int size;
        printf("Enter the size: ");
        scanf("%d",&size);
        x[i] = (double)size;

    }
    for(int i =0; i<n;i++){
        y[i]=(double)stepsforTernarySearch((int)x[i]);
    }
    RGBABitmapImageReference *imageredf = CreateRGBABitmapImageReference();
    DrawScatterPlot(imageredf,600,400,x,n,y,n);
    size_t length;
    double *pngData = ConvertToPNG(&length, imageredf->image);
    WriteTofile(pngData,length,"steps_vs_length.png");
    return 0;
}
```


00000
99999
00000
99999
00000000000000
5.00000000000000
9.00000000000000