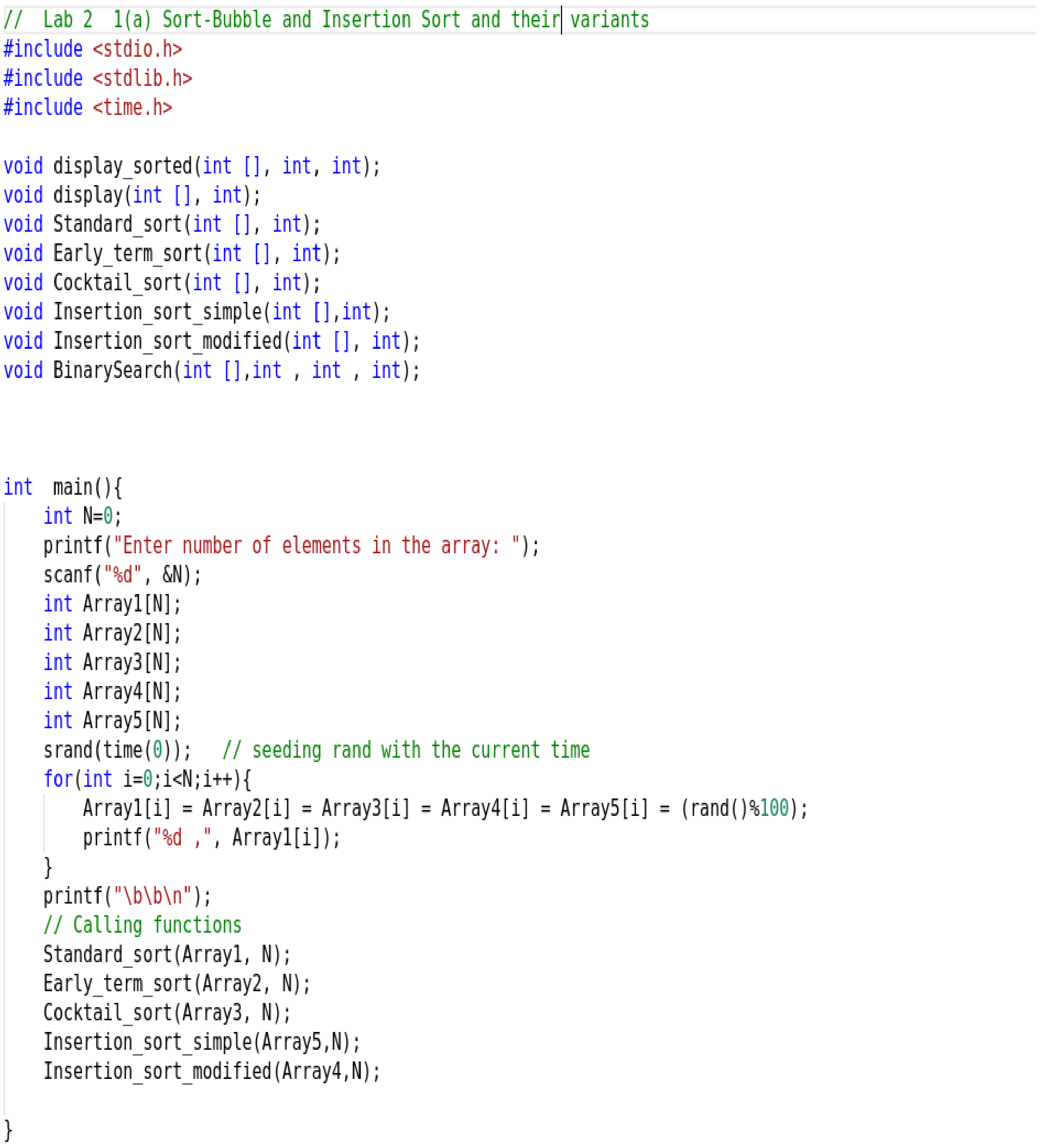
LAB-2

Name : Mohd. Zaid Ali
Faculty Number: 19COB103
Course: B. Tech
Subject : COC2910

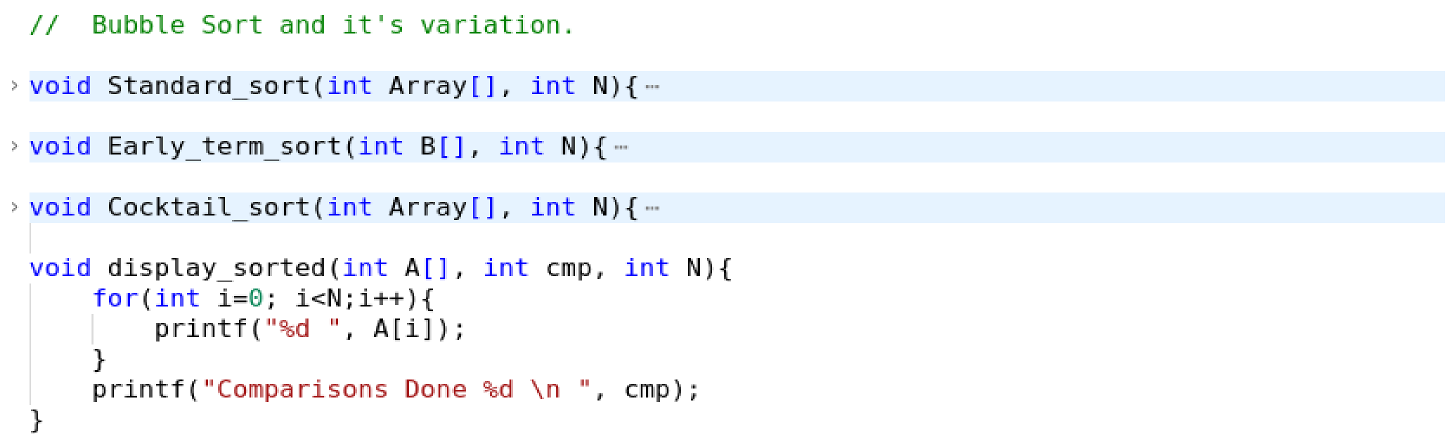
Program 1:



```c
//  Lab 2  1(a) Sort-Bubble and Insertion Sort and their variants
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void display_sorted(int [], int, int);
void display(int [], int);
void Standard_sort(int [], int);
void Early_term_sort(int [], int);
void Cocktail_sort(int [], int);
void Insertion_sort_simple(int [],int);
void Insertion_sort_modified(int [], int);
void BinarySearch(int [],int , int , int);



int  main(){
    int N=0;
    printf("Enter number of elements in the array: ");
    scanf("%d", &N);
    int Array1[N];
    int Array2[N];
    int Array3[N];
    int Array4[N];
    int Array5[N];
    srand(time(0));   // seeding rand with the current time
    for(int i=0;i<N;i++){
        Array1[i] = Array2[i] = Array3[i] = Array4[i] = Array5[i] = (rand()%100);
        printf("%d ,", Array1[i]);
    }
    printf("\b\b\n");
    // Calling functions
    Standard_sort(Array1, N);
    Early_term_sort(Array2, N);
    Cocktail_sort(Array3, N);
    Insertion_sort_simple(Array5,N);
    Insertion_sort_modified(Array4,N);

}
```

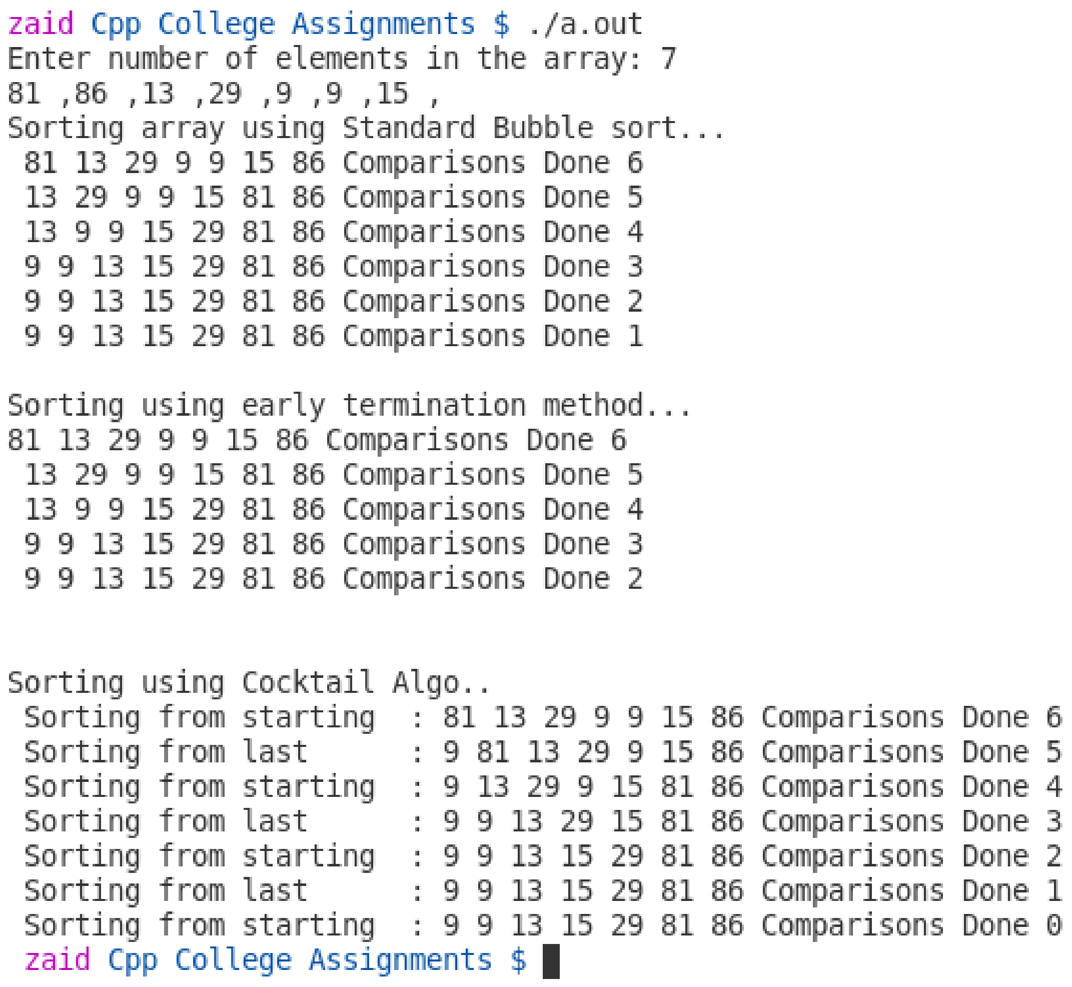
```c
int  main(){…


//  Bubble Sort and it's variation.

void Standard_sort(int Array[], int N){
    printf("Sorting array using Standard Bubble sort...\n ");
    for(int i=N-1;i>0;i--){
        int cmp =0;
        for(int j=0;j<i;j++){
            int temp =0;
            if(Array[j+1] < Array[j])
            {
                temp = Array[j+1];
                Array[j+1] = Array[j];
                Array[j] = temp;
            }
            cmp ++;
        }
        display_sorted(Array,cmp, N);
    }
    printf("\n");
}

void Early_term_sort(int B[], int N){
    printf("Sorting using early termination method...\n");
        for(int i=N-1;i>0;i--){
        int cmp =0;
        int is_comp_done = 0;
        //  needed to used int instead of the bool cause identifier not defined error...
        for(int j=0;j<i;j++){
            int temp =0;
            if(B[j+1] < B[j])
            {
                is_comp_done ++;
                temp = B[j+1];
                B[j+1] = B[j];
                B[j] = temp;
            }
            cmp ++;
        }
        display_sorted(B, cmp, N);
        if(is_comp_done == 0)
                break;
    }
    printf("\n \n");
}
```

```c
void Cocktail_sort(int Array[], int N){

    printf("Sorting using Cocktail Algo..\n ");
    int end = N-1;
    int start =0;
    int swap =1;
    while(swap==1){
                swap =0;
                int cmp = 0;
            // Strating from staring ... 0 1 2 3 ...
                printf("Sorting from starting  : ");
                for(int j=start; j< end ;j++){
                if(Array[j] > Array[j+1])
                {
                    int temp = 0;
                    temp = Array[j+1];
                    Array[j+1] = Array[j];
                    Array[j] = temp;
                    swap =1;
                }
                cmp ++;
                // if(swap==0)break;
                //since nothing is swaped all are sorted
            }
                display_sorted(Array, cmp, N);
                --end;
                cmp =0;
                if(swap ==0) break;
                printf("Sorting from last      : ") ;
            for(int j=end-1; j>=start;j--)
            {
            if(Array[j] > Array[j+1])
                {
                int temp = 0;
                temp = Array[j+1];
                Array[j+1] = Array[j];
                Array[j] = temp;
                }
            cmp ++;
            }

            ++start;
            display_sorted(Array, cmp, N);
        }
        printf("\n \n ");
    }

void display_sorted(int A[], int cmp, int N){⋯
```



```c
//  Bubble Sort and it's variation.

void Standard_sort(int Array[], int N){⋯

void Early_term_sort(int B[], int N){⋯

void Cocktail_sort(int Array[], int N){⋯

void display_sorted(int A[], int cmp, int N){
    for(int i=0; i<N;i++){
        printf("%d ", A[i]);
    }
    printf("Comparisons Done %d \n ", cmp);
}
```

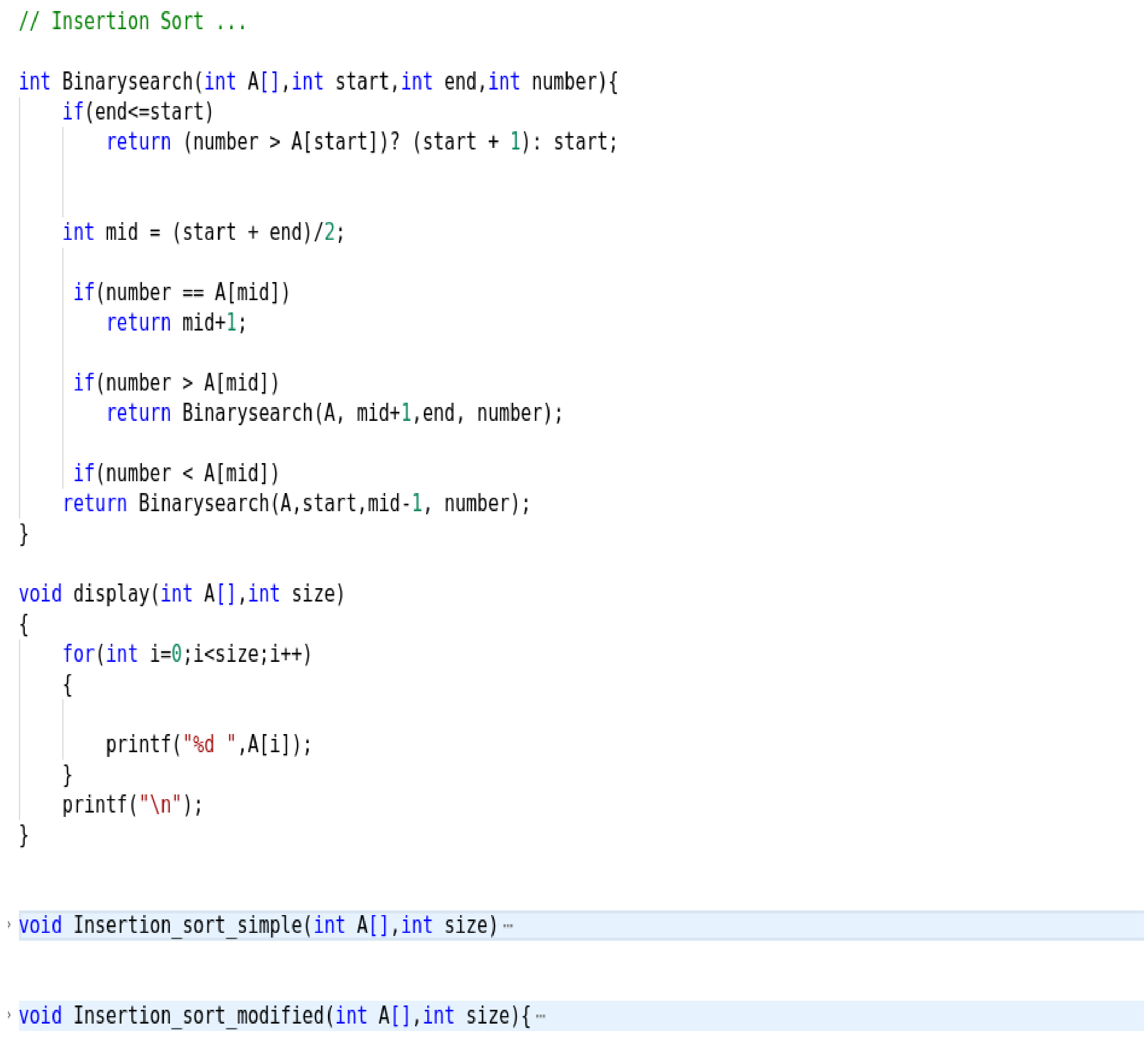
Output till the above function



```
zaid Cpp College Assignments $ ./a.out
Enter number of elements in the array: 7
81 ,86 ,13 ,29 ,9 ,9 ,15 ,
Sorting array using Standard Bubble sort...
 81 13 29 9 9 15 86 Comparisons Done 6
 13 29 9 9 15 81 86 Comparisons Done 5
 13 9 9 15 29 81 86 Comparisons Done 4
 9 9 13 15 29 81 86 Comparisons Done 3
 9 9 13 15 29 81 86 Comparisons Done 2
 9 9 13 15 29 81 86 Comparisons Done 1

Sorting using early termination method...
81 13 29 9 9 15 86 Comparisons Done 6
 13 29 9 9 15 81 86 Comparisons Done 5
 13 9 9 15 29 81 86 Comparisons Done 4
 9 9 13 15 29 81 86 Comparisons Done 3
 9 9 13 15 29 81 86 Comparisons Done 2


Sorting using Cocktail Algo..
 Sorting from starting  : 81 13 29 9 9 15 86 Comparisons Done 6
 Sorting from last      : 9 81 13 29 9 15 86 Comparisons Done 5
 Sorting from starting  : 9 13 29 9 15 81 86 Comparisons Done 4
 Sorting from last      : 9 9 13 29 15 81 86 Comparisons Done 3
 Sorting from starting  : 9 9 13 15 29 81 86 Comparisons Done 2
 Sorting from last      : 9 9 13 15 29 81 86 Comparisons Done 1
 Sorting from starting  : 9 9 13 15 29 81 86 Comparisons Done 0
 zaid Cpp College Assignments $ 
```

Program 2

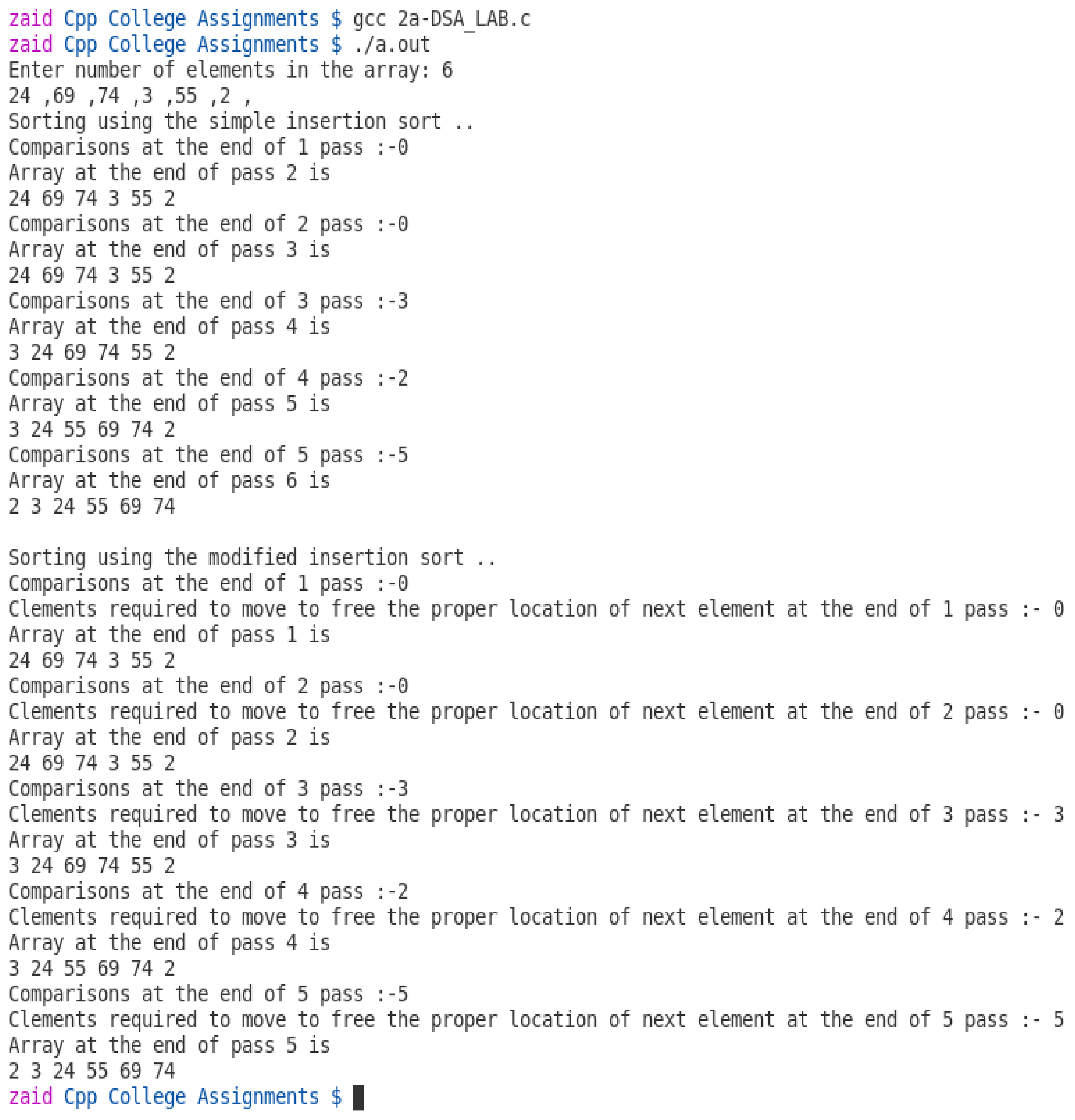
(although with in the same .c file but snapshots are shown to make them distinguishable)



```c
// Insertion Sort ...

int Binarysearch(int A[],int start,int end,int number){
    if(end<=start)
        return (number > A[start])? (start + 1): start;


    int mid = (start + end)/2;

     if(number == A[mid])
        return mid+1;

     if(number > A[mid])
        return Binarysearch(A, mid+1,end, number);

     if(number < A[mid])
    return Binarysearch(A,start,mid-1, number);
}

void display(int A[],int size)
{
    for(int i=0;i<size;i++)
    {

        printf("%d ",A[i]);
    }
    printf("\n");
}


void Insertion_sort_simple(int A[],int size)⋯


void Insertion_sort_modified(int A[],int size){⋯
void display(int A[],int size)⋯


void Insertion_sort_simple(int A[],int size)
{
    printf("Sorting using the simple insertion sort ..\n");
    int cmp=0;
    for(int i=1;i<size;i++)
    {   cmp=0;
        int value=A[i];
        int index=i;


        while(value<A[index-1]&& index>0)
        {
            A[index]=A[index-1];
            index=index-1;
            cmp++;
        }
        A[index]=value;
        printf("Comparisons at the end of %d pass :-",i);
        printf("%d\n",cmp);
        printf("Array at the end of pass %d is\n",i+1);
        display(A,size);

    }

    printf("\n");

}
```

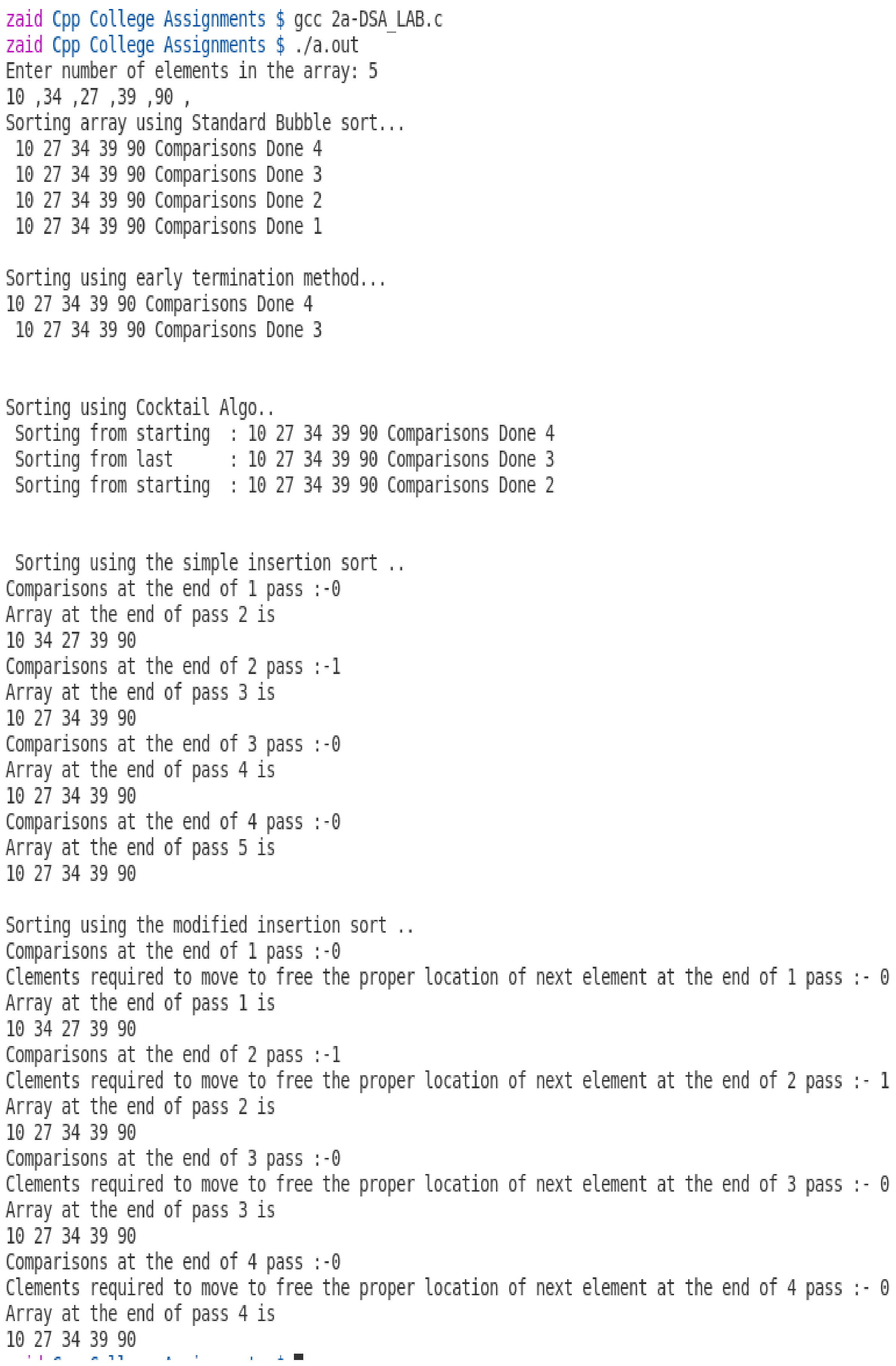
```c
void Insertion_sort_simple(int A[],int size) ⋯


void Insertion_sort_modified(int A[],int size){
    printf("Sorting using the modified insertion sort ..\n");
    int cmp=0;
    for(int i=1;i<size;i++)
    {
        cmp=0;
      int value=A[i];
      int index=i-1;
      int location=Binarysearch(A,0,index,value);
      while(index >= location)
      {
          A[index+1]=A[index];
          index=index-1;

          cmp++;
      }
      A[index+1]=value;
        printf("Comparisons at the end of %d pass :-",i);
        printf("%d\n",cmp);
        printf("Clements required to move to free the proper location of next element at the end of %d pass :- ",i);
        printf("%d\n",cmp);
        printf("Array at the end of pass %d is\n",i);
        display(A,size);
    }
}
```

Output (when only insertion sort methods called from main) :



```
zaid Cpp College Assignments $ gcc 2a-DSA_LAB.c
zaid Cpp College Assignments $ ./a.out
Enter number of elements in the array: 6
24 ,69 ,74 ,3 ,55 ,2 ,
Sorting using the simple insertion sort ..
Comparisons at the end of 1 pass :-0
Array at the end of pass 2 is
24 69 74 3 55 2
Comparisons at the end of 2 pass :-0
Array at the end of pass 3 is
24 69 74 3 55 2
Comparisons at the end of 3 pass :-3
Array at the end of pass 4 is
3 24 69 74 55 2
Comparisons at the end of 4 pass :-2
Array at the end of pass 5 is
3 24 55 69 74 2
Comparisons at the end of 5 pass :-5
Array at the end of pass 6 is
2 3 24 55 69 74

Sorting using the modified insertion sort ..
Comparisons at the end of 1 pass :-0
Clements required to move to free the proper location of next element at the end of 1 pass :- 0
Array at the end of pass 1 is
24 69 74 3 55 2
Comparisons at the end of 2 pass :-0
Clements required to move to free the proper location of next element at the end of 2 pass :- 0
Array at the end of pass 2 is
24 69 74 3 55 2
Comparisons at the end of 3 pass :-3
Clements required to move to free the proper location of next element at the end of 3 pass :- 3
Array at the end of pass 3 is
3 24 69 74 55 2
Comparisons at the end of 4 pass :-2
Clements required to move to free the proper location of next element at the end of 4 pass :- 2
Array at the end of pass 4 is
3 24 55 69 74 2
Comparisons at the end of 5 pass :-5
Clements required to move to free the proper location of next element at the end of 5 pass :- 5
Array at the end of pass 5 is
2 3 24 55 69 74
zaid Cpp College Assignments $ 
```

Output Complete:



```
zaid Cpp College Assignments $ gcc 2a-DSA_LAB.c
zaid Cpp College Assignments $ ./a.out
Enter number of elements in the array: 5
10 ,34 ,27 ,39 ,90 ,
Sorting array using Standard Bubble sort...
 10 27 34 39 90 Comparisons Done 4
 10 27 34 39 90 Comparisons Done 3
 10 27 34 39 90 Comparisons Done 2
 10 27 34 39 90 Comparisons Done 1

Sorting using early termination method...
10 27 34 39 90 Comparisons Done 4
 10 27 34 39 90 Comparisons Done 3



Sorting using Cocktail Algo..
 Sorting from starting  : 10 27 34 39 90 Comparisons Done 4
 Sorting from last      : 10 27 34 39 90 Comparisons Done 3
 Sorting from starting  : 10 27 34 39 90 Comparisons Done 2



 Sorting using the simple insertion sort ..
Comparisons at the end of 1 pass :-0
Array at the end of pass 2 is
10 34 27 39 90
Comparisons at the end of 2 pass :-1
Array at the end of pass 3 is
10 27 34 39 90
Comparisons at the end of 3 pass :-0
Array at the end of pass 4 is
10 27 34 39 90
Comparisons at the end of 4 pass :-0
Array at the end of pass 5 is
10 27 34 39 90

Sorting using the modified insertion sort ..
Comparisons at the end of 1 pass :-0
Clements required to move to free the proper location of next element at the end of 1 pass :- 0
Array at the end of pass 1 is
10 34 27 39 90
Comparisons at the end of 2 pass :-1
Clements required to move to free the proper location of next element at the end of 2 pass :- 1
Array at the end of pass 2 is
10 27 34 39 90
Comparisons at the end of 3 pass :-0
Clements required to move to free the proper location of next element at the end of 3 pass :- 0
Array at the end of pass 3 is
10 27 34 39 90
Comparisons at the end of 4 pass :-0
Clements required to move to free the proper location of next element at the end of 4 pass :- 0
Array at the end of pass 4 is
10 27 34 39 90
```