Data Structure Programs

```
#include <iostream>

using namespace std;

// function for Array Traversal

void display(int arr[] , int n) {
    for(int i = 0; i < n; i++) {
        cout<<arr[i]<<"\t";
    }
    cout<<"\n";
}


// function for Array Insertion

int Insertion(int arr[] , int n , int capacity , int element , int ind) {

    if(n >= capacity) {
        return -1;
    }
    for(int i = n; i >= ind; i--) {
       arr[i+1] = arr[i];
    }
    arr[ind] = element;
    cout<<"\n";
    return 0;
}
```

```
int main() {

    int arr[10] = {1 , 2 , 3 , 4 , 5};
    int n = 5;

    display(arr , n);

    Insertion(arr, n , 10 , 58 , 1);

     display(arr , n+1);

    return 0;
}
```

// **// Traversing : Visit each and every element of an array.**

```
// program : - >

// #include <iostream>

// using namespace std;

// int main() {

//     int len;
//     cout<<"Enter length of an Array : ";
//    cin>>len;

//    int arr[len];

//    for(int i = 0; i < len; i++) {
//        cout<<"Enter "<<i+1<<" Element : ";
//        cin>>arr[i];
//    }

//    for(int i = 0; i < len; i++) {
//      cout<<i+1<<" : "<<arr[i]<<endl;
//    }

//    return 0;
// }
```

// // Searching : Sarch element in array.

//   // 1. Linear Search : - > comparing with Each and every element.

```
// // program : - >

// #include <iostream>

//  using namespace std;



//  int linearSearch(int arr2[] , int n , int d) {

//     int i;
//     for (i = 0; i < n; i++) {
//         if(arr2[i] == d) {
//             return i;
//         }
//     }
//     return -1;
//  }

// int main() {

//    int len;
//    int dele;
//    cout<<"Enter length of an Array : ";
//     cin>>len;

//     int arr[len];

//     for (int i = 0; i < len; i++) {
//         cout<<"Enter "<<i+1<<" Element : ";
//         cin>>arr[i];
//     }

//    for (int i = 0; i < len; i++) {
//        cout<<i+1<<" : "<<arr[i]<<endl;
//    }

//    cout<<"Enter Element  to find in Array  : ";
//    cin>>dele;

//    // Call LinearSearch Function
//   int result = linearSearch(arr , len , dele);

//    if(result == -1) {
//        cout<<"Element Not Found !!!";
//    }
//    else {
//        cout<<dele<<" Found At Index : "<<result;
//    }
//    return 0;
// }
```

// 2.Binary Search : - > sort data then divide data then search element.

```
//   program : - >

 #include <iostream>

  using namespace std;



  int binarySearch(int arr2[] , int n , int d) {


  int i = 0;

    while(i <= n) {
      int  mid = (i + n) / 2;
```

```
        if(arr2[mid] ==  d) {



            return mid;
        }

        else if(d > arr2[mid]) {
            i = mid + 1;
        }



        else {
            n = mid - 1;
        }

    }

    return -1;
}



int main() {



    int len;
    int dele;
    cout<<"Enter length of an Array : ";
    cin>>len;

    int arr[len];
```

```
for (int i = 0; i < len; i++) {
   cout<<"Enter "<<i+1<<" Element : ";
   cin>>arr[i];
}

for (int i = 0; i < len; i++) {
  cout<<i+1<<" : "<<arr[i]<<endl;
}



for (int i = 0; i < len; i++) {
    for (int j = i+1; j < len; j++) {
        if(arr[i] > arr[j]) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
}

cout<<"Sorted"<<endl;

for (int i = 0; i < len; i++) {
  cout<<i+1<<" : "<<arr[i]<<endl;
}

cout<<"Enter Element  to find in Array  : ";
cin>>dele;

int result = binarySearch(arr , len , dele);
```

```
    if(result == -1) {
      cout<<"Element Not Found !!!";
   }
   else {
      cout<<dele<<" Found At Index : "<<result;
   }

    return 0;
 }
```

```
// Linear Search and Binary Search in one program : ->

#include <iostream>

using namespace std;

// Function for Linear Searching

int linearSearch(int arr[] , int n , int element) {
   for(int i = 0; i < n; i++) {
      if(arr[i] == element) {
         return i;
      }
   }
   return -1;
}
```

// Function for Binary Searching

```
int binarySearch(int arr[] , int n , int element ) {
   int lb = 0;
   int ub = n;

   while(lb<=ub) {
   int mid = ( lb + ub ) / 2;
   if(arr[mid] == element) {
      return mid;
   }
   if(arr[mid] < element) {
      lb = mid + 1;
   }
   else {
      ub = mid - 1;
   }
   }
   return -1;
}

int main() {

   int arr[] = {1 , 2 , 3 , 4 , 5 };
   int n = sizeof(arr)/sizeof(int);
   int element = 2;
  // int result = linearSearch(arr , n , element );
   int result = binarySearch(arr , n , element);
   if(result == -1) {
      cout<<"Element not found !!!";
   }
   else {
   cout<<element<<" Found at index : "<<result;
   }
   return 0;
}
```

/// 1. Bubble Sorting Program : ->

```
#include <iostream>

using namespace std;

void display(int arr[] , int n) {
    for (int i = 0; i < n; i++) {
        cout<<arr[i]<<"\t";
    }
    cout<<"\n";
}

// void bubbleSort(int arr[] , int n ) {

//     for (int i = 0; i < n-1; i++) {
//         for(int j = 0; j < n-1-i; j++) {
//             if(arr[j] > arr[j+1]) {
//                 int temp = arr[j];
//                 arr[j] = arr[j+1];
//                 arr[j+1] = temp;
//             }
//         }
//    }
// }

void bubbleSortAdaptive(int arr[] , int n ) {
    int isSorted = 0;
    for (int i = 0; i < n-1; i++) {
        cout<<"Working on pass "<<i<<endl;
        isSorted = 1;

        for(int j = 0; j < n-1-i; j++) {
            if(arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                isSorted = 0;
            }
        }
    if(isSorted == 1) {
        break;
    }
    }
}




int main() {

   int arr[] = {  1, 2 , 3 ,4 ,5 };
   int n = sizeof(arr)/sizeof(int);
   display(arr , n);
   //bubbleSort(arr , n);
  // display(arr , n);
   bubbleSortAdaptive(arr , n);
    display(arr , n);

    return 0;
}
```

// 2. Insertion sort : - > Consider Firs element as Sorted , In Insertion sort data divide into

//   two parts then it store next term in temp variable & and check conditions.

```
#include <iostream>

using namespace std;

void display(int arr[] , int n) {
    for(int i = 0; i < n; i++) {
        cout<<arr[i]<<"\t";
    }
}

void insertion_sort(int arr[] , int n ) {

    for(int i = 1; i < n; i++ ) {
        int temp = arr[i];
        int j = i - 1;

        while(j >= 0 && arr[j] > temp) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = temp;
    }


    cout<<"\n";


}


int main()
{
    int size;

    cout<<"Enter Array length : ";
    cin>>size;

    int arr[size];

    for (int i = 0; i < size; i++) {
        cin>>arr[i];
    }
    display(arr , size);
    insertion_sort(arr , size);
    display(arr , size);
    return 0;
}
```

// 3. Selection sort : - > Consider empty element as Sorted , In Selection sort it send less or minimum Number at in sorted position.

```
#include <iostream>

using namespace std;

void display(int arr[] , int n) {
    for(int i = 0; i < n; i++) {
        cout<<arr[i]<<"\t";
    }
}

void selection_sort(int arr[] , int n ) {

   int indMin;

   for (int i = 0; i < n-1; i++) {
       indMin = i;



       for(int j = i+1; j < n; j++ ) {
            if(arr[i] > arr[j]) {
                indMin = j;
            }
        }

        int temp = arr[i];
        arr[i] = arr[indMin];
        arr[indMin] = temp;
    }

    cout<<"\n";


}



int main()
{
    int size;

    cout<<"Enter Array length : ";
    cin>>size;

    int arr[size];

    for (int i = 0; i < size; i++) {
        cin>>arr[i];
    }

    display(arr , size);
    selection_sort(arr , size);
```

```
   display(arr , size);
   return 0;
}
```

// Stack Program : - >

```
 #include <iostream>

 using namespace std;

 int stack[5];
 int n = 5;
 int top =-1;

 void push(){

    int x;
    cout<<"Enter Data : ";
    cin>>x;

    if(top == n -1) {
       cout<<"Stack Is Overflow";
    }
    else {
       top++;
       stack[top] = x;
    }
 }

 void pop(){
    if(top == -1) {
        cout<<"Stack Is Underflow";
    }
    else {
        top--;

    }
}



void peek(){
    if(top == -1) {
        cout<<"Stack Is Empty";
    }
    else {
        cout<<stack[top];
    }
}

void display(){
    for(int i = top; i >= 0; i--) {
        cout<<stack[i];
    }
}

int main() {
    int choice;

    do {
        cout<<"Enter Choice - 1.Push() , 2.Pop() , 3.Peek() , 4.Display : ";
        cin>>choice;

        switch(choice) {
          case 1: push();
          break;

          case 2: pop();
          break;

          case 3: peek();
          break;

          case 4: display();
          break;

          default : cout<<"Invalid";
        }

    }while(choice != 0);

    return 0;
}
```