Lab 2

Static List

Objective:

In this lab, students will learn how to create, manuplate and display one dimension arrays. How arrays can be passed to functions and how arrays can be returned. They will also get familiarize with structures & arrays of structures. Student will also do hands-on practice to make ArrayList using integer array and static linked list.

Activity Outcomes:

The activities provide hands - on practice with the following topics

- Array Declaration and Initialization
- Array Traversal – input and output
- Passing Arrays to a Function and returning Array from function
- Creating ArrayList and related functions
- Creating StaticLinkedList and functions to manage

Instructor Note:

As a pre-lab activity, read fundamental concepts from the text book “C++ How to Program, Deitel, P. & Deitel, H., Prentice Hall, 2019”.”.

6) Useful Concepts

In C++, an array is a collection of elements of the same type placed in contiguous memory locations. Arrays provide a way to store multiple values in a single variable, which can be accessed using indices. Here's a detailed overview of arrays in C++:

Declaration and Initialization

To declare an array, you specify the type of its elements and the number of elements it can hold. For example:

```
int arr[5]; // Declares an array of 5 integers
```

You can also initialize the array at the time of declaration:

```
int arr[5] = {1, 2, 3, 4, 5}; // Array of 5 integers initialized
with values
```

If you don't specify the size, it is inferred from the number of initializers:

```
int arr[] = {1, 2, 3, 4, 5}; // Size is automatically 5
```

Accessing Elements:

Array elements are accessed using an index, with indices starting from 0. For example:

```
int firstElement = arr[0]; // Accesses the first element

int thirdElement = arr[2]; // Accesses the third element
```

Size and Iteration:

You can determine the size of an array using the `sizeof` operator. For example:

```
int size = sizeof(arr) / sizeof(arr[0]); // Calculates the number of
elements in the array
```

You can iterate through an array using a loop:

```
for (int i = 0; i < 5; ++i) {
```

```
        cout << arr[i] << " ";
    }
```

Characteristics of Arrays

1. **Fixed Size:**
   The size of an array is fixed at compile time. This means once you declare an array, its size cannot be changed during runtime.

2. **Contiguous Memory:**
   Arrays allocate memory for all their elements in a contiguous block, which allows for efficient access and iteration.
3. **Zero-Based Indexing:**
   Array indices start at 0. For an array of size `n`, valid indices range from `0` to `n-1`.
4. **Multidimensional Arrays:**
   C++ supports multidimensional arrays, such as two-dimensional arrays (matrices). For example:

```
int matrix[3][4]; // A 2D array with 3 rows and 4 columns

int matrix[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
// Initialization
```

7) Solved Lab Activites

Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
1	***5 Mins***	***Low***	***CLO-4***
2	***5 Mins***	***Low***	***CLO-4***
3	***5 Mins***	***Low***	***CLO-4***
4	***5 Mins***	***Medium***	***CLO-4***
5	***10 Mins***	***Low***	***CLO-4***
6	***15 Mins***	***Medium***	***CLO-4***
7	***10 Mins***	***Medium***	***CLO-4***
8	***20 Mins***	***Medium***	***CLO-4***

Activity 1:

Use of Arrays – array declaration, initialization, and traversal.

Solution:

```
//     First Program
   Our
   Multiple line
/*
* comment */

#include<iostream>

using namespace std;

int main() {
    int A[5]={3,4,6,7,8};
    for (int i=0; i<5;i++)
       cout<<A[i] << ",";
    return 0;
}
```

Output

3, 4, 6, 7, 8

Activity 2:

Write a function to display contents of an array.

```
#include <iostream>
using namespace std;
void display(int A[], int size) {
    for (int i=0;i<size;i++) cout << A[i];
}
int main () {
    int A[5]={3,4,6,7,8};
    display(A,5);
    return 0;
}
```

Output

3, 4, 6, 7, 8

Activity 3:

Write a function which will exchange two elements of an array.

```
#include <iostream>
using namespace std;
void swap(int A[], int i,int j) // A[i] ←→ A[j]
{
   int t=A[i];
   A[i]=A[j];
   A[j]=t;
}
void display(int A[], int size) {
    for (int i=0;i<size;i++) cout << A[i];
}
int main () {
    int A[5]={3,4,6,7,8};
    display(A,5);
    swap(A,2,4);
    display(A,5);
    return 0;
}
```

Output

3, 4, 6, 7, 8

3, 4, 8, 7, 6

Activity 4:

Creating an arraylist using simple integer array, and write functions to insert and display elements.

```
#include <iostream>
```

```
using namespace std;
const int MAX_SIZE = 10;
int A[MAX_SIZE];
int position=0;  // where the first value be inserted

void insert(int A[], int value) {
   if (position<MAX_SIZE)
          A[position++] = value;
  }

void display(int A[]) {
    for (int i=0;i<position;i++) cout << A[i];
}
int main () {
    insert(A, 3);
    insert(A, 4);
    insert(A, 6);
    insert(A, 7);
    insert(A, 8)
    display(A);
    return 0;
}
```

Output

```
3, 4, 6, 7, 8
```

Activity 5:

Write a function to delete an element from ArrayList.

```
#include <iostream>
using namespace std;
const int MAX_SIZE = 10;
```

```
int position=0;  // where the first value to be inserted
void insert(int A[],  int value) {
   if (position<MAX_SIZE)
          A[position++] = value;
  }
void remove(int A[], index i) // remove the value from i index
{
   if (i<position) {
       for (int j=i; j<position;j++)
             A[j]=A[j+1];
       position--;
     }
}

void display(int A[]) {
    for (int i=0;i<position;i++) cout << A[i];
}
int main () {
    insert(A, 33);
    insert(A, 44);
    insert(A, 56);
    insert(A, 67);
    insert(A, 78);
    remove(A, 3);
    display(A);
    return 0;
}
```

Output

```
33, 44, 56, 78
```

Activity 6:

Suppose we have two ArrayLists let say A and B, how do we can create two lists using above code.

The answer is to create two arrays, two MAX_SIZE variables and two position variables, such as

int A[MAX_SIZE_A], int B[MAX_SIZE_B] etc.

In above example, we created an array to store values, a maximum variable which stores how many values can be stored, and the third variable position to keep track of the position in array where next value will be inserted.

To group these items together we need to use either classes or structures; As per our course requirement we will use the structures to group things together.

To define composite types we use structures, one of the common examples of composite types is date, where three integers make up a date structure;

```
#include <iostream>
using namespace std;

const int MAX_SIZE = 10;
struct ArrayList
 {
    int Element[MAX_SIZE];
    int position=0;
  };

void insert(ArrayList *List, int value) {
   if (List->position<MAX_SIZE)
          List->Element[List->position++] = value;
  }
void remove(ArrayList *List, int index) {
   if (index<List->position) {
       for (int j=index; j<List->position;j++)
            List->Element[j]=List->Element[j+1];
```

```
        List->position--;
      }
}
void display(ArrayList *List) {
    for (int i=0;i<List->position;i++)
          cout << List->Element[i];
}
int main () {
    ArrayList A, B;
    insert(&A, 3);
    insert(&A, 5);
    insert(&B, 6);
    insert(&B, 8);
    display(&A);
    display(&B);
    return 0;
}
```

Output:

3, 5

6, 8

Activity 7:

In previous two activites we used remove function, which removes the content of ith position value with the next (i+1)th value, and respectively all the value upto last element are shifted one step backward.

This approach is considered slow – ie. If we remove first element from list we need to shift rest of the list. To tackle this issue and to make list more effienct we can simple mark that element as any –ve number so that can be considered as deleted.

```
#include <iostream>
using namespace std;
```

```
const int MAX_SIZE = 10;
struct ArrayList
 {
    int Element[MAX_SIZE];
    int position=0;
  };

void insert(ArrayList *List, int value) {
   if (List->position<MAX_SIZE)
          List->Element[List->position++] = value;
  }
void remove(ArrayList *List, int index) {
   if (index<List->position) {
       List->Element[j]= -9999; // this is considered deleted
     }
}
void display(ArrayList *List) {
    for (int i=0;i<List->position;i++)
          if (List->Element[i]!=-9999)
               cout << List->Element[i];
}
int main () {
    ArrayList A, B;
    insert(&A, 3);
    insert(&A, 5);
    insert(&B, 6);
    insert(&B, 8);
    display(&A);
    display(&B);
    return 0;
}
```

8) Graded Lab Tasks

Lab Task 1

use a simple array to store the marks of 10 students from quiz 1 of data structure course, the obtained marks are considered out of 15. Write function to compute average marks for the entire class.

Lab Task 2

Write a program using float ArrayList to store temperature of a week, and write functions to find week maximum temprature.

Lab Task 3

By using the concept of static linked list, write functions for the following operations.