Ahsania Mission University of Science & Technology

Lab Report

Course Code: CSE 2202

Course Title: Computer Algorithm Sessional

Experiment No: 01

Experiment Date: 05.02.25

Submitted By:

Mehrin Nusrat Chowdhury

Roll: 1012320005101026

1st Batch, 2nd Year, 1st Semester

Department of Computer Science and Engineering

Ahsania Mission University of Science & Technology

Submitted To:

Md. Fahim Faisal

Lecturer,

Department of Computer Science and Engineering

Faculty of Engineering, Ahsania Mission University of Science & Technology

Submission Date: 19.02.25

Task 01: A C++ program that will merge two arrays.

Theory:

Merging two arrays involves combining the elements of two separate arrays into a single array. The process starts by copying the elements of the first array and the elements of the second array into the new merged array. This task helps to understand array traversal and the basics of combining datasets.

Code:

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

int main()
{
   int arr1[100] = {2, 4, 6}; // First array
   int size1 = 3;
   int arr2[100] = {8, 10, 12, 14}; // Second array
   int size2 = 4;

   int mergedSize = size1 + size2;
   int merged[mergedSize];

   //Copying first array
   for(int i = 0 ; i < size1 ; i++)
   {
      merged[i] = arr1[i];
   }
   //Copying second array
   for(int i = 0 ; i < size2 ; i++)
   {
      merged[size1 + i] = arr2[i];
   }

   //Print
   cout << "The merged array: ";
   for(int i = 0 ; i < mergedSize ; i++)
   {
      cout << merged[i] << " ";
   }
   cout << "\n";
   return 0;
}
```

Output:

```
The merged array: 2 4 6 8 10 12 14

Process returned 0 (0x0)   execution time : 0.064 s
Press any key to continue.
```

Conclusion:

The merging process successfully combines the elements of two arrays into one. This task reinforces the understanding of array manipulation, providing a fundamental basis for more complex data handling and algorithms.

Task 02: A C++ program that will calculate the sum of an array.

Theory:

Summing the elements of an array is a fundamental operation in programming. The task involves reading an array of integers and then calculating the total sum by iterating through each element. This helps in understanding loops and accumulation operations which are common in data processing.

Code:

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

int main()
{
   int n;
   cout << "Enter the number of elements of array A: ";
   cin >> n;

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

   int sum = 0;

   for(int i = 0 ; i < n ; i++)
   {
      sum = sum + A[i];
   }

   cout << "Sum = " << sum;

   return 0;
}
```

Output:

```
Enter the number of elements of array A: 5
8 2 4 1 4
Sum = 19
Process returned 0 (0x0)   execution time : 7.329 s
Press any key to continue.
```

Conclusion:

The task demonstrates the basic operation of iterating through an array and accumulating the sum of its elements. It underscores the importance of loop constructs and provides a solid foundation for more advanced data aggregation techniques.

Task 03: A C++ program that will find the maximum in an array.

Theory:

Finding the maximum value in an array involves scanning through all the elements and comparing them to determine the highest value. This task illustrates the use of conditional statements and loops to identify specific characteristics within a dataset.

Code:

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

int main()
{
   int testCase;
   cout << "Enter test cases: ";
   cin >> testCase;
   for(int i = 1 ; i < testCase ; i++)
   {
      int n;
      cout << "\nEnter the number of elements: ";
      cin >> n;
      int arr[n];
      cout << "Enter the elements: ";
      cout << "\n";
      for(int i = 0 ; i < n ; i++)
      {
         cin >> arr[i];
      }

      int max_height = arr[0];
      for(int i = 1 ; i < n ; i++)
      {
         if(arr[i] > max_height)
         {
            max_height = arr[i];
         }
      }
      cout << "\nHighest Height: " << max_height;

   }
   return 0;
}
```

Output:

```
Enter test cases: 2

Enter the number of elements: 6
Enter the elements:
167 162 152 151 161 150

Highest Height: 167
Process returned 0 (0x0)   execution time : 25.771 s
Press any key to continue.
```

Conclusion:

The task of finding the maximum value in an array highlights the process of iterating through data and applying conditional logic. It serves as a practical exercise in array manipulation and reinforces the concept of identifying key attributes within a dataset.

Task 04: A C++ Program to Optimize Array Values by Making the Minimum the Maximum.

Theory:

This task involves manipulating an array to make its minimum value become the maximum value through a series of operations. Each operation allows setting an element to any value between 1 and 100. The aim is to determine the minimum number of operations required. This task teaches optimization techniques and problem-solving strategies.

Code:

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

int main()
{
   int t;
   cout << "Enter test cases: ";
   cin >> t;

   while(t--)
   {
      int n;
      cout << "\nEnter the number of elements: ";
      cin>>n;

      int a[n];
      cout << "Elements: ";
      for(int i = 0 ; i < n ; i++)
      {
         cin >> a[i];
      }

      int min_val = a[0];
      for(int i = 1 ; i < n ; i++)
      {
         if(a[i] < min_val)
         {
            min_val = a[i];
         }
```

```
        }

        int Count = 0;
        for(int i = 0 ; i < n ; i++)
        {
           if(a[i] > min_val)
           {
              Count++;
           }
        }
        cout << "\nOutput: " << Count << "\n";

    }
    return 0;
}
```

Output:

```
Enter test cases: 3

Enter the number of elements: 2
Elements: 1 2

Output: 1

Enter the number of elements: 4
Elements: 2 2 3 4

Output: 2

Enter the number of elements: 1
Elements: 1

Output: 0

Process returned 0 (0x0)   execution time : 20.141 s
Press any key to continue.
```

Conclusion:

The task provides insight into optimization problems where a series of operations are performed to achieve a desired outcome. It emphasizes the importance of strategic thinking and efficient algorithm design in solving real-world problems.