C++ Assignments | Arrays - 3 | Week 5

1.Count the number of triplets whose sum is equal to the given value x.

2.Find the factorial of a large number.

3.Find the first non-repeating element in the array .

4.Check if an array is a subset of another .

Output:-

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

int countTriplets(int arr[], int size, int x) {
    int count = 0;
    for (int i = 0; i < size - 2; i++) {
        for (int j = i + 1; j < size - 1; j++) {
            for (int k = j + 1; k < size; k++) {
                if (arr[i] + arr[j] + arr[k] == x) {
                    count++;
                }
            }
        }
    }
    return count;
}

int main() {
    int arr[] = {1, 4, 6, 3, 9, 2, 7};
    int x = 12;
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << "Number of triplets with sum " << x << " is: " << countTriplets(arr, size, x) << endl;
    return 0;
}
```

—-Number of triplets with sum 12 is: 2

2.

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

void multiply(vector<int> &result, int x) {
  int carry = 0;
  for (int i = 0; i < result.size(); i++) {
    int prod = result[i] * x + carry;
    result[i] = prod % 10;
    carry = prod / 10;
  }
  while (carry) {
    result.push_back(carry % 10);
    carry /= 10;
  }
}
```

```
void factorial(int n) {
    vector<int> result(1, 1);
    for (int i = 2; i <= n; i++) {
        multiply(result, i);
    }
    for (int i = result.size() - 1; i >= 0; i--) {
        cout << result[i];
    }
    cout << endl;
}

int main() {
    int n = 50;
    cout << "Factorial of " << n << " is: ";
    factorial(n);
    return 0;
}
```

——Factorial of 50 is: 30414093201713378043612608166064768844377641568960512000000000000

3.

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

int firstNonRepeating(int arr[], int size) {
    unordered_map<int, int> count;
    for (int i = 0; i < size; i++) {
        count[arr[i]]++;
    }
    for (int i = 0; i < size; i++) {
        if (count[arr[i]] == 1) {
            return arr[i];
        }
    }
    return -1; // If no non-repeating element found
}

int main() {
    int arr[] = {9, 4, 9, 6, 7, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    int result = firstNonRepeating(arr, size);
    if (result != -1) {
        cout << "The first non-repeating element is: " << result << endl;
    } else {
        cout << "There is no non-repeating element." << endl;
    }
    return 0;
}
```

——The first non-repeating element is: 6

4.

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

bool isSubset(int arr1[], int size1, int arr2[], int size2) {
    unordered_set<int> set;
    for (int i = 0; i < size1; i++) {
        set.insert(arr1[i]);
    }
    for (int i = 0; i < size2; i++) {
        if (set.find(arr2[i]) == set.end()) {
            return false;
        }
    }
    return true;
}

int main() {
    int arr1[] = {11, 1, 13, 21, 3, 7};
    int arr2[] = {11, 3, 7, 1};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    if (isSubset(arr1, size1, arr2, size2)) {
        cout << "arr2 is a subset of arr1." << endl;
    } else {
        cout << "arr2 is not a subset of arr1." << endl;
    }
    return 0;
}
```

```
——arr2 is a subset of arr1.
```