Data Structure Programs in C++

```
// 1. Binary Search
#include <iostream>
using namespace std;
int binarySearch(int arr[], int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == key) return mid;
        else if (arr[mid] < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}
int main() {
    int arr[] = {11, 22, 33, 44, 55};
    int key = 44;
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << binarySearch(arr, n, key);
    return 0;
}

// 2. Bubble Sort
#include <iostream>
using namespace std;
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = 7;
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(arr[j], arr[j + 1]);
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}

// 3(a). Missing Number - Linear Time
#include <iostream>
using namespace std;
int main() {
    int arr[] = {1, 2, 3, 5};
    int n = 5, sum = 0;
    for (int i = 0; i < n - 1; i++) sum += arr[i];
    int total = n * (n + 1) / 2;
    cout << total - sum;
    return 0;
}

// 3(b). Missing Number - Binary Search
#include <iostream>
using namespace std;
int main() {
    int arr[] = {1, 2, 3, 5};
    int low = 0, high = 3, mid;
    while (low <= high) {
        mid = (low + high) / 2;
        if (arr[mid] != mid + 1 && arr[mid - 1] == mid) break;
```

```
        else if (arr[mid] == mid + 1) low = mid + 1;
        else high = mid - 1;
    }
    cout << mid + 1;
    return 0;
}

// 4. String Operations
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cctype>
using namespace std;
int main() {
    string s1 = "Hello", s2 = "World";
    s1 += s2;
    cout << s1 << endl;

    string rev = s1;
    reverse(rev.begin(), rev.end());
    cout << rev << endl;

    string s3 = "Beautiful";
    string res = "";
    for (char c : s3)
        if (string("aeiouAEIOU").find(c) == string::npos)
            res += c;
    cout << res << endl;

    string str[] = {"banana", "apple", "grape", "mango", "cherry"};
    sort(str, str + 5);
    for (auto &x : str) cout << x << endl;

    char ch = 'A';
    cout << (char)tolower(ch);
    return 0;
}

// 5. Matrix Storage
#include <iostream>
using namespace std;
int main() {
    int n = 3;
    int diag[3] = {1, 2, 3};
    int tri[7] = {1, 4, 0, 2, 5, 3, 6};
    int lower[6] = {1, 2, 3, 4, 5, 6};
    int upper[6] = {1, 2, 3, 4, 5, 6};
    int sym[6] = {1, 2, 3, 4, 5, 6};
    cout << "Stored successfully";
    return 0;
}

// 6. Sparse Matrix - Transpose
#include <iostream>
using namespace std;
int main() {
    int a[3][3] = {{1, 0, 0}, {0, 2, 0}, {0, 0, 3}};
    int t[3][3];
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            t[j][i] = a[i][j];
    for (int i = 0; i < 3; i++) {
```

```
        for (int j = 0; j < 3; j++)
            cout << t[i][j] << " ";
        cout << endl;
    }
    return 0;
}

// 7. Count Inversions
#include <iostream>
using namespace std;
int main() {
    int arr[] = {1, 20, 6, 4, 5}, n = 5, count = 0;
    for (int i = 0; i < n - 1; i++)
        for (int j = i + 1; j < n; j++)
            if (arr[i] > arr[j]) count++;
    cout << count;
    return 0;
}

// 8. Count Distinct Elements
#include <iostream>
using namespace std;
int main() {
    int arr[] = {1, 2, 2, 3, 4, 4, 5};
    int n = 7, count = 0;
    for (int i = 0; i < n; i++) {
        bool flag = false;
        for (int j = 0; j < i; j++)
            if (arr[i] == arr[j]) flag = true;
        if (!flag) count++;
    }
    cout << count;
    return 0;
}
```