## **ASSIGNMENT 2&3[DSA]**

Submitted by :Disha(1024030180)

Q1

## LINEAR SEARCH

```
#include <iostream>
   using namespace std;
 4 - int linearSearch(int arr[], int size, int target) {
       for (int i = 0; i < size; i++) {
            if (arr[i] == target)
 6
                return i;
 8
 9
        return -1;
10 }
11
12 - int main() {
        int arr[] = {34, 78, 12, 56, 89, 23};
        int size = sizeof(arr) / sizeof(arr[0]);
14
       int target = 56;
15
       int idx = linearSearch(arr, size, target);
16
17
        if (idx != -1)
18
           cout << "Element found at index " << idx << endl;</pre>
19
            cout << "Element not found in the array." << endl;</pre>
20
21
        return 0:
22 }
23
```

## **BINARY SEARCH**

```
1 #include <iostream> //binary search
  2 using namespace std;
  3
  4- int binarySearch(int arr[], int size, int target) {
  5
        int left = 0;
  6
         int right = size - 1;
  7 -
         while (left <= right) {
  8
             int mid = left + (right - left) / 2; // safe against overflow
  9
             if (arr[mid] == target)
 10
                 return mid:
 11
             else if (arr[mid] < target)</pre>
 12
                 left = mid + 1;
 13
             else
 14
                right = mid - 1;
 15
 16
         return -1;
 17 }
 18
 19 - int main() {
 20
         int arr[] = {12, 23, 34, 56, 78, 89};
         int size = sizeof(arr) / sizeof(arr[0]);
 21
 22
         int target = 56;
 23
         int idx = binarySearch(arr, size, target);
         if (idx != -1)
 24
```

```
wniie (lett <= right) {
            int mid = left + (right - left) / 2; // safe against overflow
9
            if (arr[mid] == target)
10
                return mid;
11
            else if (arr[mid] < target)</pre>
12
                left = mid + 1;
13
            else
14
                right = mid - 1;
15
        }
16
        return -1;
17 }
18
19 - int main() {
20
        int arr[] = {12, 23, 34, 56, 78, 89};
21
        int size = sizeof(arr) / sizeof(arr[0]);
        int target = 56;
22
        int idx = binarySearch(arr, size, target);
23
24
        if (idx != -1)
25
            cout << "Element found at index " << idx << endl;</pre>
26
            cout << "Element not found in the array." << endl;</pre>
27
28
        return 0;
29 }
30
```

Q2

```
1 #include <iostream>
2 using namespace std;
 4 - void bubbleSort(int arr[], int size) {
 5 +
        for (int i = 0; i < size - 1; i++) {
 6 +
            for (int j = 0; j < size - i - 1; j++) {
7 -
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
 8
 9
                    arr[j] = arr[j + 1];
10
                    arr[j + 1] = temp;
11
                }
12
            }
13
        }
14 }
15
16 - void printArray(int arr[], int size) {
        for (int i = 0; i < size; i++) {
17 -
            cout << arr[i] << " ";
18
19
        cout << endl;</pre>
20
21 }
22
23 - int main() {
        int arr[1 = {64. 34. 25. 12. 22. 11. 90}:
```

```
14 }
15
16 - void printArray(int arr[], int size) {
17 -
        for (int i = 0; i < size; i++) {
             cout << arr[i] << " ";
18
19
20
        cout << endl;
21
   }
22
23 - int main() {
        int arr[] = {64, 34, 25, 12, 22, 11, 90};
24
25
        int size = sizeof(arr) / sizeof(arr[0]);
26
27
        cout << "Original array: ";
28
        printArray(arr, size);
29
        bubbleSort(arr, size);
30
31
        cout << "Sorted array:</pre>
32
33
        printArray(arr, size);
34
35
        return 0;
36 }
37
```

Q3

```
1 #include <iostream>
2 using namespace std;
3
4- int findMissingLinear(int arr[], int size) {
        for (int i = 0; i < size; i++) {
            if (arr[i] != i + 1) {
6 +
7
                return i + 1;
8
            }
9
        }
10
        return size + 1; // Missing number is n
11 }
12
13 - int main() {
14
        int arr[] = {1, 2, 3, 4, 6, 7, 8}; // n = 8, size = 7
15
        int size = sizeof(arr) / sizeof(arr[0]);
16
        cout << "Missing number (linear): " << findMissingLinear(arr, size) <<</pre>
            endl;
        return 0;
17
18 }
19
```

```
1 #include <iostream>
2 using namespace std;
 4 - int findMissingBinary(int arr[], int size) {
        int left = 0, right = size - 1;
        while (left <= right) {</pre>
 6 +
            int mid = (left + right) / 2;
 7
            // Found first mismatch:
 8
 9
            if (arr[mid] != mid + 1 &&
10 -
                (mid == 0 \mid \mid arr[mid - 1] == mid)) {
11
                return mid + 1;
12
            }
13 -
            if (arr[mid] == mid + 1) {
                // Everything up to mid is correct-go right
14
15
                left = mid + 1;
16 -
            } else {
                // Mismatch occurs on or before mid-go left
17
                right = mid - 1;
18
19
20
        }
        return -1; // Shouldn't happen if input guarantees one missing
21
22 }
23
24 - int main() {
```

```
9
            1T (arr[m10] != m10 + 1 &&
10 -
                 (mid == 0 \mid | arr[mid - 1] == mid)) {
11
                 return mid + 1;
12
            }
13 -
            if (arr[mid] == mid + 1) {
14
                 // Everything up to mid is correct—go right
                 left = mid + 1;
15
16 -
             } else {
17
                 // Mismatch occurs on or before mid-go left
18
                 right = mid - 1;
19
             }
20
         }
        return -1; // Shouldn't happen if input guarantees one missing
21
22 }
23
24 - int main() {
         int arr[] = {1, 2, 3, 4, 6, 7, 8}; // n = 8, size = 7
26
         int size = sizeof(arr) / sizeof(arr[0]);
27
         cout << "Missing number (binary): " << findMissingBinary(arr, size) <<</pre>
28
         return 0;
29
30
31
```

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 - int main() {
        string s1 = "Hello, ";
6
7
        string s2 = "World!";
8
        string result = s1 + s2; // or s1.append(s2);
        cout << "Concatenated string: " << result << endl;</pre>
9
        return 0;
10
11 }
12
```

В

```
-- -
1 #include <iostream>
2 #include <string>
3 #include <algorithm>
4 using namespace std;
5
6 - int main() {
       string s = "Hello, world!";
7
8
       reverse(s.begin(), s.end());
       cout << "Reversed string: " << s << endl;</pre>
9
10
       return 0;
11 }
12
13
```

С

```
[] Share
                                                                Run
main.cpp
   #include <iostream>
   using namespace std;
 4 - bool isVowel(char c) {
 5
        c = tolower(c);
        return c=='a' || c=='e' || c=='i' || c=='o' || c=='u';
 6
 7
 8
9 - int main() {
10
        char str[100];
11
        cout << "Enter string: ";</pre>
12
        cin.getline(str, 100);
13
14
       int j = 0;
15 -
        for (int i = 0; str[i] != '\0'; i++) {
            if (!isVowel(str[i])) {
16 -
17
                str[j++] = str[i];
18
19
20
        str[j] = ' \0';
21
        cout << "After deleting vowels: " << str << endl;</pre>
22
23
        return 0;
24 }
```

```
1 #include <iostream>
 2 #include <vector>
 3 #include <string>
 4 #include <algorithm>
 5 using namespace std;
 7 - int main() {
        vector<string> arr = {"banana", "apple", "cherry", "date"};
 9
        sort(arr.begin(), arr.end());
10
        cout << "Sorted strings:\n";</pre>
11 -
        for (const auto &s : arr) {
12
            cout << s << endl;</pre>
13
       return 0;
14
15 }
16
17
```

Ε

```
1 #include <iostream>
2 using namespace std;
3
4 - int main() {
 5
        string s = "HeLLo, WoRLD!";
        for (char &c : s) {
 6 -
            if (c >= 'A' && c <= 'Z') {
 7 -
 8
                c = c + ('a' - 'A'); // Add 32
9
            }
10
        }
11
        cout << "Lowercase string: " << s << endl;</pre>
12
        return 0;
13 }
14
15
16
```

```
1 #include <iostream>
 2 using namespace std;
 3
 4 - int main() {
        int n = 4; // Example size
        int diag[4] = {1, 2, 3, 4}; // Diagonal Matrix -> only n elements
 7
        int tri[10]; // Tri-diagonal Matrix -> 3n-2 elements
        int lower[10]; // Lower triangular -> n(n+1)/2 elements
 9
10
        int upper[10]; // Upper triangular -> n(n+1)/2 elements
        int sym[10]; // Symmetric -> n(n+1)/2 elements
11
12
13
        cout << "Space Efficient Storage Demonstrated" << endl;</pre>
14
        return 0:
15 }
```

Q6

```
4 - int main() {
        int r1=3, c1=3, r2=3, c2=3;
        // Triplet form: row col value
        int sparse1[4][3] = \{\{0,0,1\},\{0,2,2\},\{1,1,3\},\{2,0,4\}\}\};
        int sparse2[3][3] = \{\{0,1,5\},\{1,2,6\},\{2,2,7\}\};
10
        cout << "Transpose of sparse1:" << endl;</pre>
11
        for(int i=0; i<4; i++){
12 -
            cout << sparse1[i][1] << " " << sparse1[i][0] << " " <<</pre>
13
                 sparse1[i][2] << endl;
14
15
        cout << "Addition of sparse1 and sparse2:" << endl;</pre>
16
        for(int i=0;i<4;i++) cout << sparse1[i][0] << " " << sparse1[i][1] << "
17
             " << sparse1[i][2] << endl;
        for(int i=0;i<3;i++) cout << sparse2[i][0] << " " << sparse2[i][1] << "</pre>
             " << sparse2[i][2] << endl;
19
        cout << "Multiplication not shown fully (too complex), but idea:</pre>
             multiply triplets row-wise * column-wise" << endl;
21
        return 0;
22
```

```
1 #include <iostream>
 2 using namespace std;
 3
 4 - int main() {
  5
       int n = 5;
       int A[5] = {2, 4, 1, 3, 5};
  6
  7
       int count = 0;
 8
 9 +
     for(int i=0;i<n;i++){
       for(int j=i+1;j<n;j++){
 10 -
              if(A[i] > A[j]) count++;
 11
 12
       }
 13
        }
 14
        cout << "Number of inversions = " << count << endl;</pre>
 15
 16
        return 0;
 17 }
```