|  |
| --- |
| 1. Write a function that prints the numbers from 1 to 100. But for multiples of three, print "Fizz" instead of the number, and for the multiples of five, print "Buzz". For numbers that are multiples of both three and five, print "FizzBuzz". |
| function fizzBuzz() {  for (let i = 1; i <= 100; ++i) {  if (i % 3 === 0 && i % 5 === 0) {  console.log("FizzBuzz");  } else if (i % 3 === 0) {  console.log("Fizz");  } else if (i % 5 === 0) {  console.log("Buzz");  } else {  console.log(i);  }  }  }  // Run the function  fizzBuzz(); |
|  |

**Global Trend Programming Profile Assessment Questions and Solution**

|  |
| --- |
| 1. Write a function that takes a string input representing a simple arithmetic expression (only addition and subtraction) and returns the result |
| #include <iostream>  #include <string>  using namespace std;  int evaluateExpression(const string &expression)  {    int result = 0;    int number = 0;    char op = '+';    for (size\_t i = 0; i < expression.size(); ++i)    {      char ch = expression[i];      if (isdigit(ch))      {        number = number \* 10 + (ch - '0');      }      if (!isdigit(ch) && !isspace(ch) || i == expression.size() - 1)      {        if (op == '+')        {          result += number;        }        else if (op == '-')        {          result -= number;        }        op = ch;        number = 0;      }    }    return result;  }  int main()  {    string expression;    cout << "Enter a simple arithmetic expression (only addition and subtraction): ";    getline(cin, expression);    int result = evaluateExpression(expression);    cout << "The result is: " << result << endl;    return 0;  } |
|  |

|  |
| --- |
| 1. Write a function that takes a nested array and returns a flattened array. |
| #include <iostream>  #include <vector>  using namespace std;  // Recursive function to flatten the nested vector  void flatten(const vector<vector<int>> &arr, vector<int> &result)  {    for (const auto &elem : arr)    {      if (elem.empty())        continue;      if (elem.size() == 1 && !elem[0].empty())      {        result.push\_back(elem[0]);      }      else      {        flatten(elem, result);      }    }  }  int main()  {    // Define a nested vector    vector<vector<int>> nestedArray = {{1}, {2, 3}, {4}, {5, {6}}};    // Vector to hold the flattened array    vector<int> flattenedArray;    // Flatten the nested vector    flatten(nestedArray, flattenedArray);    // Print the flattened array    for (int num : flattenedArray)    {      cout << num << " ";    }    return 0;  } |
|  |

|  |
| --- |
| 1. Write a function that checks if two given strings are anagrams of each other |
| #include <bits/stdc++.h>  using namespace std;  bool isAnagram(string a, string b)  {    if (a.length() != b.length())    {      return false;    }    unordered\_map<char, int> Map;    for (int i = 0; i < a.length(); i++)    {      Map[a[i]]++;    }    for (int i = 0; i < b.length(); i++)    {      if (Map.find(b[i]) != Map.end())      {        Map[b[i]] -= 1;      }      else      {        return false;      }    }    for (auto items : Map)    {      if (items.second != 0)      {        return false;      }    }    return true;  }  int main()  {    string str1;    string str2;    cout << "Enter the first string: ";    cin >> str1;    cout << "Enter the Second string: ";    cin >> str2;    if (isAnagram(str1, str2))      cout << "True"           << endl;    else      cout << "False"           << endl;  } |
|  |

|  |
| --- |
| 1. Write a function that takes an array and returns a new array with duplicates removed |
| #include <iostream>  #include <vector>  #include <bits/stdc++.h>  using namespace std;  vector<int> removeDuplicates(vector<int> &arr)  {    sort(arr.begin(), arr.end());    int j = 0;    for (int i = 0; i < arr.size(); i++)    {      if (i == 0 || arr[i] != arr[i - 1])      {        arr[j] = arr[i];        j++;      }    }    arr.resize(j);    return arr;  }  int main()  {    vector<int> arr;    int size;    cout << "Enter the number of elements: ";    cin >> size;    for (int i = 0; i < size; ++i)    {      int num;      cout << "Enter element " << i + 1 << ": ";      cin >> num;      arr.push\_back(num);    }    arr = removeDuplicates(arr);    cout << "Unique array: ";    for (int num : arr)    {      cout << num << " ";    }    cout << endl;    return 0;  } |
|  |

|  |
| --- |
| 1. Write a function that takes a string and capitalizes the first letter of each word in the string |
| #include <iostream>  #include <sstream>  #include <algorithm>  using namespace std;  string titleCase(string text)  {    transform(text.begin(), text.end(), text.begin(), ::tolower);    stringstream ss(text);    string word;    string result;    while (ss >> word)    {      if (!word.empty())      {        word[0] = toupper(word[0]);        result += word + " ";      }    }    if (!result.empty() && result[result.size() - 1] == ' ')    {      result.pop\_back();    }    return result;  }  int main()  {    cout << "Enter a string: ";    string userInput;    getline(cin, userInput);    string result = titleCase(userInput);    cout << result << endl;    return 0;  } |
|  |

|  |
| --- |
| 1. Write a function that generates the first n numbers of the Fibonacci sequence |
| #include <iostream>  #include <vector>  using namespace std;  vector<int> fibonacci(int n)  {    if (n <= 1)    {      return vector<int>(n, 0);    }    vector<int> fibonacci\_sequence = {0, 1};    for (int i = 2; i < n; ++i)    {      int next\_number = fibonacci\_sequence[i - 1] + fibonacci\_sequence[i - 2];      fibonacci\_sequence.push\_back(next\_number);    }    return fibonacci\_sequence;  }  int main()  {    int n;    cout << "Enter the number of Fibonacci numbers: ";    cin >> n;    vector<int> fibonacci\_sequence = fibonacci(n);    cout << "Fibonacci sequence: ";    for (int num : fibonacci\_sequence)    {      cout << num << " ";    }    cout << endl;    return 0;  } |
|  |

|  |
| --- |
| 8) Implement a simple HashMap class with put, get, and remove methods |
| #include <iostream>  #include <vector>  #include <unordered\_map>  using namespace std;  struct KeyValuePair  {    int key;    string value;  };  class HashMap  {  public:    HashMap(int size = 10) : map\_(size) {}    void put(int key, const string &value)    {      KeyValuePair kv = {key, value};      map\_[key] = kv;    }    string get(int key)    {      if (map\_.count(key) > 0)      {        return map\_[key].value;      }      else      {        return "";      }    }    bool remove(int key)    {      if (map\_.count(key) > 0)      {        map\_.erase(key);        return true;      }      else      {        return false;      }    }  private:    unordered\_map<int, KeyValuePair> map\_;  };  int main()  {    HashMap myMap;    myMap.put(1, "apple");    myMap.put(2, "banana");    myMap.put(3, "cherry");    cout << "Value for key 2: " << myMap.get(2) << endl;    cout << "Removed key 1: " << (myMap.remove(1) ? "Success" : "Fail") << endl;    cout << "Value for key 1: " << myMap.get(1) << endl;    return 0;  } |
|  |

|  |
| --- |
| 1. Write a function that filters out even numbers from an array. |
| #include <iostream>  #include <vector>  using namespace std;  vector<int> filtereven(vector<int> arr, int n)  {    vector<int> result;    for (auto i : arr)    {      if (i % 2 != 0) // Checking for odd numbers (filtering out even numbers)        result.push\_back(i);    }    return result;  }  int main()  {    int n;    cout << "Enter the size of array: ";    cin >> n;    vector<int> arr(n); // Resize the vector to size n    cout << "Enter the elements in the array: ";    for (int i = 0; i < n; i++)    {      cin >> arr[i];    }    cout << "Original array: ";    for (int i = 0; i < n; i++)    {      cout << arr[i] << " ";    }    cout << endl;    vector<int> result = filtereven(arr, n);    cout << "Filtered array (only odd numbers): ";    for (auto i : result)    {      cout << i << " ";    }    cout << endl;    return 0;  } |
|  |

|  |
| --- |
| 1. Write a function that converts a given string to title case (capitalizing the first letter of each word). |
| #include <iostream>  #include <sstream>  #include <algorithm>  using namespace std;  string titleCase(string text)  {    transform(text.begin(), text.end(), text.begin(), ::tolower);    stringstream ss(text);    string word;    string result;    while (ss >> word)    {      if (!word.empty())      {        word[0] = toupper(word[0]);        result += word + " ";      }    }    if (!result.empty() && result[result.size() - 1] == ' ')    {      result.pop\_back();    }    return result;  }  int main()  {    cout << "Enter a string: ";    string userInput;    getline(cin, userInput);    string result = titleCase(userInput);    cout << result << endl;    return 0;  } |
|  |