**Q.1: How to reverse a string?**

**Ans.:**The user will input a string and the method should return the reverse of that string

* input: hello, output: olleh
* input: hello world, output: dlrow olleh

1. **internal** static **void** ReverseString(string str)
2. {
3. char[] charArray = str.ToCharArray();
4. **for** (int i = 0, j = str.Length - 1; i < j; i++, j--)
5. {
6. charArray[i] = str[j];
7. charArray[j] = str[i];
8. }
9. string reversedstring = **new** string(charArray);
10. Console.WriteLine(reversedstring);
11. }

**Q.2: How to find if the given string is a palindrome or not?**

**Ans.:**The user will input a string and we need to print “*Palindrome*” or “*Not Palindrome*” based on whether the input string is a palindrome or not.

* input: madam, output: Palindrome
* input: step on no pets, output: Palindrome
* input: book, output: Not Palindrome

if we pass an integer as string parameter then also this method will give the correct output

* input: 1221, output: Palindrome

1. **internal** static **void** chkPalindrome(string str)
2. {
3. bool flag = **false**;
4. **for** (int i = 0, j = str.Length - 1; i < str.Length / 2; i++, j--)
5. {
6. **if** (str[i] != str[j])
7. {
8. flag = **false**;
9. **break**;
10. }
11. **else**
12. flag = **true**;
13. }
14. **if** (flag)
15. {
16. Console.WriteLine("Palindrome");
17. }
18. **else**
19. Console.WriteLine("Not Palindrome");
20. }

**Q.3: How to reverse the order of words in a given string?**

**Ans.:**The user will input a sentence and we need to reverse the sequence of words in the sentence.

* input: Welcome to Csharp corner, output: corner Csharp to Welcome

1. **internal** static **void** ReverseWordOrder(string str)
2. {
3. int i;
4. StringBuilder reverseSentence = **new** StringBuilder();
5. int Start = str.Length - 1;
6. int End = str.Length - 1;
7. **while** (Start > 0)
8. {
9. **if** (str[Start] == ' ')
10. {
11. i = Start + 1;
12. **while** (i <= End)
13. {
14. reverseSentence.Append(str[i]);
15. i++;
16. }
17. reverseSentence.Append(' ');
18. End = Start - 1;
19. }
20. Start--;
21. }
22. **for** (i = 0; i <= End; i++)
23. {
24. reverseSentence.Append(str[i]);
25. }
26. Console.WriteLine(reverseSentence.ToString());
27. }

**Q.4: How to reverse each word in a given string?**

**Ans.:**The user will input a sentence and we need to reverse each word individually without changing its position in the sentence.

* input: Welcome to Csharp corner, output: emocleW ot prahsC renroc

1. **internal** static **void** ReverseWords(string str)
2. {
3. StringBuilder output = **new** StringBuilder();
4. List<char> charlist = **new** List<char>();
5. **for** (int i = 0; i < str.Length; i++)
6. {
7. **if** (str[i] == ' ' || i == str.Length - 1)
8. {
9. **if** (i == str.Length - 1)
10. charlist.Add(str[i]);
11. **for** (int j = charlist.Count - 1; j >= 0; j--)
12. output.Append(charlist[j]);
13. output.Append(' ');
14. charlist = **new** List<char>();
15. }
16. **else**
17. charlist.Add(str[i]);
18. }
19. Console.WriteLine(output.ToString());
20. }

**Q.5: How to count the occurrence of each character in a string?**

**Ans.:**The user will input a string and we need to find the count of each character of the string and display it on console. We won’t be counting space character.

* input: hello world;

output: h – 1

e – 1

l – 3

o – 2

w – 1

r – 1

d – 1

1. **internal** static **void** Countcharacter(string str)
2. {
3. Dictionary<char, int> characterCount = **new** Dictionary<char, int>();
4. **foreach** (var character in str)
5. {
6. **if** (character != ' ')
7. {
8. **if** (!characterCount.ContainsKey(character))
9. {
10. characterCount.Add(character, 1);
11. }
12. **else**
13. {
14. characterCount[character]++;
15. }
16. }
17. }
18. **foreach** (var character in characterCount)
19. {
20. Console.WriteLine("{0} - {1}", character.Key, character.Value);
21. }
22. }

**Q.6: How to remove duplicate characters from a string?**

**Ans.:**The user will input a string and the method should remove multiple occurrences of characters in the string

* input: csharpcorner, output: csharpone

1. **internal** static **void** removeduplicate(string str)
2. {
3. string result = string.Empty;
4. **for** (int i = 0; i < str.Length; i++)
5. {
6. **if** (!result.Contains(str[i]))
7. {
8. result += str[i];
9. }
10. }
11. Console.WriteLine(result);
12. }

**Q.7: How to find all possible substring of a given string?**

**Ans.:**This is a very frequent interview question. Here we need to form all the possible substrings from input string, varying from length 1 to the input string length. The output will include the input string also.

* input: abcd , output : a ab abc abcd b bc bcd c cd d

1. **internal** static **void** findallsubstring(string str)
2. {
3. **for** (int i = 0; i < str.Length; ++i)
4. {
5. StringBuilder subString = **new** StringBuilder(str.Length - i);
6. **for** (int j = i; j < str.Length; ++j)
7. {
8. subString.Append(str[j]);
9. Console.Write(subString + " ");
10. }
11. }
12. }

**Q.8: How to perform Left circular rotation of an array?**

**Ans.:**The user will input an integer array and the method should shift each element of input array to its Left by one position in circular fashion. The logic is to iterate loop from Length-1 to 0 and swap each element with last element.

* input: 1 2 3 4 5, output: 2 3 4 5 1

1. **internal** static **void** RotateLeft(int[] array)
2. {
3. int size = array.Length;
4. int temp;
5. **for** (int j = size - 1; j > 0; j--)
6. {
7. temp = array[size - 1];
8. array[array.Length - 1] = array[j - 1];
9. array[j - 1] = temp;
10. }
11. **foreach** (int num in array)
12. {
13. Console.Write(num + " ");
14. }
15. }

**Q.9: How to perform Right circular rotation of an array?**

**Ans:** The user will input an integer array and the method should shift each element of input array to its Right by one position in circular fashion. The logic is to iterate loop from 0 to Length-1 and swap each element with first element

* input: 1 2 3 4 5, output: 5 1 2 3 4

1. **internal** static **void** RotateRight(int[] array)
2. {
3. int size = array.Length;
4. int temp;
5. **for** (int j = 0; j < size - 1; j++)
6. {
7. temp = array[0];
8. array[0] = array[j + 1];
9. array[j + 1] = temp;
10. }
11. **foreach** (int num in array)
12. {
13. Console.Write(num + " ");
14. }
15. }

**Q.10: How to find if a positive integer is a prime number or not?**

**Ans.:**The user will input a positive integer and the method should output “*Prime*” or “*Not Prime*” based on whether the input integer is a prime number or not.

The logic is to find a positive integer less than or equal to the square root of input integer. If there is a divisor of number that is less than the square root of number, then there will be a divisor of number that is greater than square root of number. Hence, we have to traverse till the square root of number.

The time complexity of this function is O(√N) because we traverse from 1 to √N.

* input: 20, output: Not Prime
* input: 17, output: Prime

1. static **void** Main(string[] args)
2. {
3. **if** (FindPrime(47))
4. {
5. Console.WriteLine("Prime");
6. }
7. **else**
8. {
9. Console.WriteLine("Not Prime");
10. }
11. Console.ReadLine();
12. }
13. **internal** static bool FindPrime(int number)
14. {
15. **if** (number == 1) **return** **false**;
16. **if** (number == 2) **return** **true**;
17. **if** (number % 2 == 0) **return** **false**;
18. var squareRoot = (int)Math.Floor(Math.Sqrt(number));
19. **for** (int i = 3; i <= squareRoot; i += 2)
20. {
21. **if** (number % i == 0) **return** **false**;
22. }
23. **return** **true**;
24. }

**Q.11: How to find the sum of digits of a positive integer?**

**Ans.:**The user will input a positive integer and the method should return the sum of all the digits in that integer.

* input: 168, output: 15

1. **internal** static **void** SumOfDigits(int num)
2. {
3. int sum = 0;
4. **while** (num > 0)
5. {
6. sum += num % 10;
7. num /= 10;
8. }
9. Console.WriteLine(sum);
10. }

**Q.12: How to find second largest integer in an array using only one loop?**

**Ans.:** The user will input an unsorted integer array and the method should find the second largest integer in the array.

* input: 3 2 1 5 4, output: 4

1. **internal** static **void** FindSecondLargeInArray(int[] arr)
2. {
3. int max1 = int.MinValue;
4. int max2 = int.MinValue;
5. **foreach** (int i in arr)
6. {
7. **if** (i > max1)
8. {
9. max2 = max1;
10. max1 = i;
11. }
12. **else** **if** (i >= max2 && i != max1)
13. {
14. max2 = i;
15. }
16. }
17. Console.WriteLine(max2); ;
18. }

**Q.13: How to find third largest integer in an array using only one loop?**

**Ans.:** The user will input an unsorted integer array and the method should find the third largest integer in the array.

* input: 3 2 1 5 4, output: 3

1. **internal** static **void** FindthirdLargeInArray(int[] arr)
2. {
3. int max1 = int.MinValue;
4. int max2 = int.MinValue;
5. int max3 = int.MinValue;
6. **foreach** (int i in arr)
7. {
8. **if** (i > max1)
9. {
10. max3 = max2;
11. max2 = max1;
12. max1 = i;
13. }
14. **else** **if** (i > max2 && i != max1)
15. {
16. max3 = max2;
17. max2 = i;
18. }
19. **else** **if** (i > max3 && i != max2 && i != max1)
20. {
21. max3 = i;
22. }
23. }
24. Console.WriteLine(max3); ;
25. }

**Q.14: How to convert a two-dimensional array to a one-dimensional array?**

**Ans.:** The user will input a 2-D array (matrix) and we need to convert it to a 1-D array. We will create 1-D array column wise.

* input: { { 1, 2, 3 }, { 4, 5, 6 } }, output: 1 4 2 5 3 6

1. **internal** static **void** MultiToSingle(int[,] array)
2. {
3. int index = 0;
4. int width = array.GetLength(0);
5. int height = array.GetLength(1);
6. int[] single = **new** int[width \* height];
7. **for** (int y = 0; y < height; y++)
8. {
9. **for** (int x = 0; x < width; x++)
10. {
11. single[index] = array[x, y];
12. Console.Write(single[index] + " ");
13. index++;
14. }
15. }
16. }

This question can also be asked to form 1-D array row wise. In this case just swap the sequence of thefor loops as below. The output will be 1 2 3 4 5 6 for the input matrix mentioned above.

1. **for** (int x = 0; x < width; x++ )
2. {
3. **for** ( int y = 0; y < height; y++)
4. {
5. single[index] = array[x, y];
6. Console.Write(single[index] + " ");
7. index++;
8. }
9. }

**Q.15: How to convert a one-dimensional array to a two-dimensional array?**

**Ans.:** The user will input 1-D array along with the number of rows and columns. The method should convert this 1-D array to a 2-D array(matrix) of given row and column. We will create matrix row wise.

* input: {1, 2, 3, 4, 5, 6} ,2 ,3

output: 1 2 3

4 5 6

1. **internal** static **void** SingleToMulti(int[] array, int row, int column)
2. {
3. int index = 0;
4. int[,] multi = **new** int[row, column];
5. **for** (int y = 0; y < row; y++)
6. {
7. **for** (int x = 0; x < column; x++)
8. {
9. multi[y, x] = array[index];
10. index++;
11. Console.Write(multi[y, x] + " ");
12. }
13. Console.WriteLine();
14. }
15. }

**Q.16: How to find the angle between hour and minute hands of a clock at any given time?**

**Ans.:** The user will input hour and minute of the time and the method should give the angle between the hour hand and minute hand at that given time.

* input: 9 30, output: The angle between hour hand and minute hand is 105 degrees
* input: 13 30, output: The angle between hour hand and minute hand is 135 degrees

The logic is to find the difference in the angle of hour and minute hand from the position of 12 O Clock, when angle between them is zero. Each hour on the clock represent an angle of 30 degrees (360 divided by 12). Similarly, each minute on the clock will represent an angle of 6 degrees (360 divided by 60) and the angle for an hour will increase as the minutes for that hour increases. So, our code will be

1. **internal** static **void** FindAngleinTime(int hours, int mins)
2. {
3. double hourDegrees = (hours \* 30) + (mins \* 30.0 / 60);
4. double minuteDegrees = mins \* 6;
5. double diff = Math.Abs(hourDegrees - minuteDegrees);
6. **if** (diff > 180)
7. {
8. diff = 360 - diff;
9. }
10. Console.WriteLine("The angle between hour hand and minute hand is {0} degrees", diff);
11. }

**Q.17: Explain Bubble Sort Algorithm In C#**

This algorithm follows the concept of iterating through the array from the first index to the last index and comparing adjacent elements and then swapping them if they appear in the wrong order. i.e. If the next element is smaller than the current element, they are swapped.

Time Complexity of bubble sort is **O(n²)**.

Read [Bubble Sort Algorithm In C#](https://ankitsharmablogs.com/bubble-sort-algorithm-in-c-sharp/) to learn more.

**Q.18: Explain Quick Sort Algorithm In C#**

This Algorithm selects an element as pivot element from the given array and partitions the array around it such that, Left side of pivot contains all the elements that are less than the pivot element. The right side contains all elements that are greater than the pivot element. Let P be the index of pivot after partitioning the array. Then, the left subarray(start to P-1) and right subarray(P+1 to end) are sorted recursively to get the final sorted array as output.

The worst case time complexity of this algorithm is **O(N²)**. The best case and average case time complexity of this algorithm is **O(NLogN)**.

Read [Quick Sort Algorithm In C#](https://ankitsharmablogs.com/quick-sort-algorithm-in-c-sharp/) to learn more.

**Q.19: Explain Merge Sort Algorithm In C#**

This algorithm works as follows.

* Divide the unsorted array of size N into N subarrays having single element each.
* Take two adjacent subarrays and merge them to form a sorted subarray having 2 elements.Now we have N/2 subarrays of size 2.
* Repeat the process until a single sorted array is obtained.

In all the three cases (worst, average, best), the time complexity of Merge sort is **O(NLogN)**

Read [Merge Sort Algorithm In C#](https://ankitsharmablogs.com/merge-sort-algorithm-in-c-sharp/) to learn more.

**Q. 20: Explain Insertion Sort Algorithm In C#**

Insertion sort compares the current element with largest value in the sorted array. If the current element is smaller then the algorithm finds its correct position in sorted array and moves the element to that position otherwise if the current element is greater then it leaves the element in its place and moves on to next element.

To place the element in its correct position in sorted array, all the elements larger than the current element is shifted one place ahead.Thus the sorted array will grow at each iteration.

Every element is compared to every other element of sorted array. Hence complexity of Insertion sort is **O(n²)**.

Read [Insertion Sort Algorithm In C#](https://ankitsharmablogs.com/insertion-sort-algorithm-in-c-sharp/) to learn more.

**Q. 21: Explain Selection Sort Algorithm In C#**

This algorithm follows the concept of dividing the given array into two subarrays.

1. The subarray of sorted elements
2. The subarray of unsorted elements

The algorithm will find the minimum element in the unsorted subarray and put it into its correct position in the sorted subarray.

Read [Selection Sort Algorithm In C#](https://ankitsharmablogs.com/selection-sort-algorithm-in-c-sharp/) to learn more.

Time complexity of Selection Sort as **O(n²).**

**Q. 22: Explain Binary Search In C#**

Binary search is an efficient and commonly used searching algorithm.This algorithm works only on sorted sets of elements. So if the given array is not sorted then we need to sort it before applying Binary search.

This algorithm searches a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.if found return the index of matched element , else return -1.

Read [Searching Algorithms In C#](https://ankitsharmablogs.com/searching-algorithms-in-c-sharp/) to learn more.

The array to be searched is reduced by half in every iteration. Hence time complexity of the Binary search is **O(LogN)**.