

# Group 3 Manual

MENTORED BY  
Anshul, Ujjwal and Manas

**Compiled on 23/03/2021 at 16:30:00**

# Contents

<b>1</b>	<b>Assignment 1</b>	<b>1</b>
1.1	System Setup . . . . .	1
<b>2</b>	<b>Switching from C to C++</b>	<b>2</b>
2.1	Why so serious? . . . . .	2
2.2	C++ style strings . . . . .	2
<b>3</b>	<b>Assignment 2</b>	<b>4</b>
3.1	How much do you know? . . . . .	4
3.1.1	Level 1 . . . . .	4
3.1.2	Level 2 . . . . .	4
3.1.3	Level 3 . . . . .	4
<b>4</b>	<b>Baby Steps</b>	<b>5</b>
4.1	Making I/O fast in C++ . . . . .	5
4.2	A basic template file . . . . .	5
4.3	Frequently seen numbers . . . . .	6
4.4	Common Terminology . . . . .	6
<b>5</b>	<b>Two Pointers</b>	<b>7</b>
5.1	Editorial for Zeroes problem . . . . .	7

# Chapter 1

## Assignment 1

### 1.1 System Setup

- Install an editor/IDE
- Choose a Language
- Install a compiler

## Chapter 2

# Switching from C to C++

### 2.1 Why so serious?

---

```
1  #include <stdio.h>
2
3  int main() {
4
5      int n;
6      long long m;
7      char s[1000];
8
9      scanf("%d%lld%s", &n, &m, &s);
10
11     printf("%d %lld %s\n", n, s);
12
13     return 0;
14 }
```

---

becomes

---

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5
6      int n;
7      long long m;
8      char s[1000];
9
10     \ a new data structure for strings which makes life easy
11     string ss;
12
13     cin >> n >> m >> s >> ss;
14
15     cout << n << ' ' << m << ' ' << s << ' ' << ss << '\n';
16
17     return 0;
18 }
```

---

## 2.2 C++ style strings

In C++, the **string** data type is very similar to character arrays, the only thing is here we do not have a NULL terminator. You get the following new functionalities:

```
string s = "Hello", s1 = "abcd", s2 = "abcd";
```

- append characters :

```
s += '2';
```

- append string :

```
s += ", how are you?"
```

- compare strings :

```
if(s1 == s2) cout << "Strings are equal!!\n";
```

- indexing operator[] :

```
for(int i = 0; i < s.length(); i++) { cout << s[i]; } cout << '\n';
```

will print all the characters of the string. But that is not how you print a string, you directly do

```
cout << s << '\n';
```

# Chapter 3

## Assignment 2

### 3.1 How much do you know?

The aim is to gauge your level of expertise so that we can provide everyone with what they want without wasting anybody's time. Don't cheat or lie, it will defeat the purpose of the assignment.

#### 3.1.1 Level 1

1. [Fabonacci](#)
2. [Watermelon](#)

#### 3.1.2 Level 2

1. [Zeroes to the end](#) If you do it in  $O(n^2)$  we won't mind, but leetcode might give a TLE.
2. [Count Primes](#)

#### 3.1.3 Level 3

1. [Zeroes to the end](#) IMP: In a single linear  $O(n)$  traversal
2. [Convert to base 2](#) (because that BE course was not a joke)

Adding the entry to the google sheet


If you solved 1 from L1; 1, 2 from L2 and none from L3, you write it as “**10 11 00**”.

Someone who solves all will write “**11 11 11**” and will color his box green.

Partial solvers color their box blue.

Those who tried but solved none will color their's red.

This will be followed for all the future assignments. This assignment needs to be done within next 4 days, and then we will have a little discussion if required.

Indenting the code is very important, I personally follow the K&R style but you can choose Google, LLVM or clang. Use your left pinky finger to hit the  key.

# Chapter 4

## Baby Steps

### 4.1 Making I/O fast in C++

These two lines of code can make your C++ runtimes faster by 50%, you only notice the difference on large testcases.

```
ios_base::sync_with_stdio(false);
cin.tie(NULL), cout.tie(NULL);
```

### 4.2 A basic template file

A template is a piece of code you can copy everytime you solve some problem.

If the problem has no test cases:

---

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define fastIO ios_base::sync_with_stdio(0 && cin.tie(0) && cout.tie(0));
5
6  typedef long long ll;
7
8  int main() {
9      fastIO;
10
11      // insert your code here ....
12
13      return 0;
14 }
```

---

If the problem has test cases:

---

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define fastIO ios_base::sync_with_stdio(0 && cin.tie(0) && cout.tie(0));
5
6  typedef long long ll;
7
8  // this solve() function can be treated exactly like main()
9  int solve() {
```

```

10
11      // insert your code here
12
13      return 0;
14  }
15
16  int main() {
17      fastIO;
18
19      int t;
20      cin >> t;
21      while(t--) {
22          solve();
23      }
24
25      return 0;
26  }

```

---

### 4.3 Frequently seen numbers

There are a few numbers like  $10^7$ ,  $10^9 + 7$  (1000000007) , etc. which you will frequently find in problems, we will discuss what they mean over here.

- You can run  $10^7$  to  $10^8$  iterations in around one second, so mostly you will see that they give size  $n$  of array equal to  $10^5$  and number of test cases of the order of  $10^3$  or  $10^2$ .
- $10^9 + 7$  (1000000007) is provided as a huge prime number you will need to take a modulo with of your answers in combinatorics type problems, where the answer can be so large that it will overflow the 64 bit integer (long long) too! You will need to know the modulo arithmetic rules for solving such problems. 998244353 is another prime number which might be found in a few problems.
- $10^{-9} \leq x \leq 10^9$ , means you need to use **int** for storing  $x$ , anything larger than that goes inside the 64 bit **long long**, which will mostly be mentioned as  $x \leq 10^{18}$ .

### 4.4 Common Terminology

I have added the ones which always confuse me, will add more as they come to my mind.

- **substring:** A continuous piece cut from a string. For e.g. “abcd” is a substring of “aaabcccdeee” but “abcde” is not
- **subsequence:** A discontinuous piece of an array or string. In the above example “abcde” is a subsequence of “aaabcccdeee”.
- **subarray:** It is the **array** version of **substring**.
- **segment:** It is another name for a subarray.



# Chapter 5

## Two Pointers

### 5.1 Editorial for Zeroes problem

---

```
1  class Solution {
2  public:
3      void moveZeroes(vector<int>& nums) {
4
5          int n = nums.size();
6
7          int src = 0, dest = 0;
8
9          // src is the pointer which will always point to
10         // non zero numbers, whereas dest will point to
11         // the first occurrence of zero at any moment
12         // we pick the non-zero number from src and swap it with dest
13
14         while(1) {
15
16             // increment dest until the numbers are non-zero
17             while(dest < n && nums[dest] != 0)
18                 dest++;
19
20             // this is common sense that src must always be greater than dest
21             if(src < dest) src = dest + 1;
22
23             // increment src until the it is zero
24             while(src < n && nums[src] == 0) src++;
25
26             // a check to ensure that src is within bounds
27             // this is also the place where the code will break
28             if(src >= n) break;
29
30             // this is the inbuilt swap function of c++
31             swap(nums[src], nums[dest]);
32
33             dest++, src++;
34
35         }
```

```
36
37     return;
38 }
39 };
```

---

You might feel that it is not  $O(n)$  but  $O(n^2)$ , so I say make a mini test case and simulate it using the above code on paper.