Some Useful Short tricks for Competitive Programming (C++)

Note: If you want to be good at competitive programming you must be familiar with the **STL** library. First do learn about STL **containers** and **algorithms**. Here is link to learn STL
https://www.geeksforgeeks.org/the-c-standard-template-library-stl/
http://www.cplusplus.com/reference/stl/

This is the best book for short tricks-

1. **Tips & Secrets for professionals(C++)** (500 pages)
The above book is of 500 pages and it will be quite difficult to read. But I prefer going through this pdf you should once through this book.

2.**Competitive Programmer's Handbook(C++)** (296 pages)
Another book which everyone should prefer. Every Competitive Programmer follow this

Here are some web links which you can also refer
1. C++ tips and tricks (Codeforces)
2. C++ tricks (Codeforces)
3. C++ tricks for competitive programming (for C++ 11)
4. Some useful C++ tricks for beginners in Competitive Programming
5. Bit Tricks for Competitive Programming
6. Logarithm tricks for Competitive Programming
7. Common mistakes to be avoided in Competitive Programming in C++ | Beginners

Now I have extracted some best tricks from all of this which everyone should prefer

1. **Checking if the number is even or odd without using the % operator:**

Instead of using
if(a%2!=0) cout<<”odd”;

You can use
if(a&1) cout<<”odd”;

2. **Fast Multiplication or Division by 2**

Instead of using
n=n*2
n=n/2

You can use
n=n<<1
n=n>>1

3. **Avoiding use of strlen():**

// Use of strlen() can be avoided by:
for (i=0; s[i]; i++)
{
}
// loop breaks when the character array ends.

4.**Get rid of those includes!**

instead of individual standard headers like `<string>`, `<iostream>` and `<vector>`. It ruins portability and fosters terrible habits.

You can use only one header file which includes everything
`#include<bits/stdc++.h>`

5. **Avoiding use of make pair**

Instead of using this

```
pair<int, int> p;
p = make_pair(3, 4);
```

You can use this

```
pair<int, int> p;
p = {3, 4};
```

6. **Use of Tuple**

Instead of using this

`pair<int, pair<int, int>>`

You can use this

`tuple<int, int, int>`

7. **Use of Auto Keyword**

Instead of using this

vector<int> lks(n)
for(int i=0;i<n;i++){
cout<<lks[i]<<” “;
}

You can Use

vector<int> lks(n)
for(auto x: lks){
cout<<x<<” “;
}

NOTE: One of the useful trick to print the element of vector is use of macro
#define print(x) for(auto it:x) cout<<it<<' '; cout<<"\n";

And then whenever you need to print the vector just write
print(lks);

8. Avoiding multiple line in for loop

Instead of using this

vector<int> lks[100];
for(int i=0;i<n;i++){
cin>>a>>b;
lks[a].push_back(b);
lks[b].push_back(a);
}

You can use this
for(int i=0;i<n;i++) cin>>a>>b, lks[a].push_back(b), lks[b].push_back(a);

9. Use of Inbuilt function

There are lots of inbuilt function which make your code simple

1. __gcd(x,y); //for gcd of two number
2. accumulate(arr[0], arr[0]+n, 0) //for sum of array

And many more swap, _builtin_popcount(R), _builtin_clz(R) etc..

10. **Calculating the number of digits directly:**

To calculate number of digits in a number, instead of looping you can efficiently use log

```
Number of digits in N =floor(log10(N)) + 1;
```

11. Use of emplace_back() instead of push_back()

Instead of push_back() in STL emplace_back can be used because it is much faster and instead of allocating memory somewhere else and then appending it directly allocates memory in the container.

12. **The Iota Algorithm**

The algorithm iota() creates a range of sequentially increasing values, as if by assigning an initial value to *first, then incrementing that value using prefix ++. In the following listing, iota() assigns the consecutive values {10, 11, 12, 13, 14} to the array arr, and {‘a’, ‘b’, ‘c’} to the char array c[].

```
int a[5] = {0};
char c[3] = {0};

// changes a to {10, 11, 12, 13, 14}
iota(a, a+5, 10);
iota(c, c+3, 'a'); // {'a', 'b', 'c'}
```

13. Use of all(x)

Define a macro

```
#define all(x) (x).begin(), (x).end()
```

Now sorting a vector looks like `sort(all(vec))` instead of `sort(vec.begin(), vec.end())`.

14..Using Conditional Operators

```
// Without conditional operators
if (a & 1) { // when a is odd
  a=a*2;
}
else{
  a=a+1;
}   // With conditional operators

a&1?a*=2:a+=1;
```

15. Using of scanf

Instead of using this

Int arr[100];

```
scanf("%d", arr[n + i]); // since t is an array
```

Use this

```
scanf("%d", t + n + i);
```

16. Use of memset

Sometimes we need to set the value of array or vector with some value or sometime we have to clear the initialised value to set them with some 0 value.
Then we can use memset for that purpose

Int arr[100];
memset(arr, 0, sizeof(arr)) // to set all the value 0 to whole array elements
memset(arr, -1, sizeof(arr)) // to set all the value -1 to whole array elements

NOTE: Whenever you have to set the value again and again in your code, then just define the macro and use whenever needed
#define clr(x) memset(x, 0, sizeof(x))

Just write
vector<int> lks[100]
clr(lks) // the whole vector elements will set to value 0.