Competitive Programming Template and Snippets

- Anikate Koul

Goal

- To understand the characteristics and importance of a good competitive programming template
- To learn the different techniques to create and modify snippets.

Importance of a Good Template

- A good competitive programming template helps in setting up a standard boilerplate code before you start coding the actual logic.
- This helps in saving time, especially during contests.

Example template

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int t;
    cin >> t;
    while (t--)
    {
        // code here
    }
    return 0;
}
```

Importance of a Good Template

- Also, having pre-written and tested snippets for algorithms that require heavy implementation eliminates the risk of silly mistakes when the time pressure is very high.

Example Snippet

```cpp
void bubble_sort(vector<int> &arr)
{
    int n = arr.size();
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}
```

Although,

- Having a large template with a lot of pre-written functions is not a must for competitive programming.
- CP is more about testing your **problem solving ability** than having a macro defined for every possible data structure.
- Many strong coders (eg. jiangly and SSRS_) do not use templates for most of their solutions.
- Therefore, it is essential to understand that having a template is completely a personal choice and not a necessity.

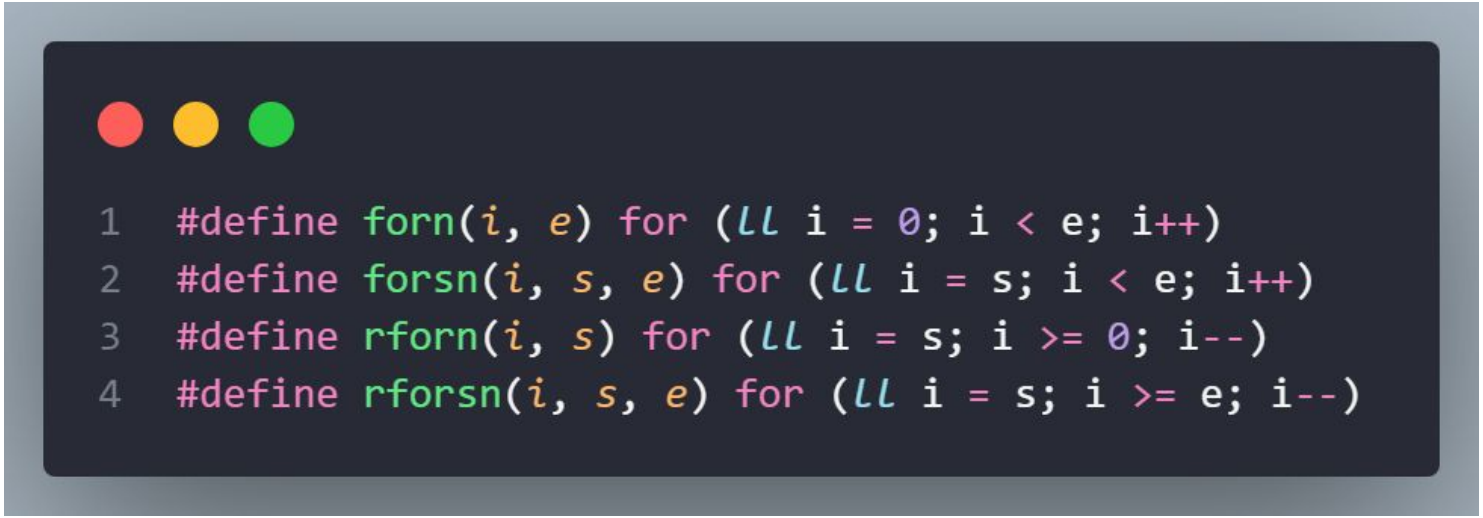
Characteristics of a good template

- Provides shorthand macros for long and commonly used data structures.
- Provides shorthand macros for standard snippets like loops.

What are macros?

- Macros are preprocessor directives in C++ used to define constants, expressions, or reusable code snippets.
- They are processed before compilation by the preprocessor.
- **#define** is used to declare constants or expressions.
- **typedef** is used to create an alias for a data type to improve readability and reusability.

Example macros



```
#define forn(i, e) for (ll i = 0; i < e; i++)
#define forsn(i, s, e) for (ll i = s; i < e; i++)
#define rforn(i, s) for (ll i = s; i >= 0; i--)
#define rforsn(i, s, e) for (ll i = s; i >= e; i--)
```

Example macros

```
typedef long long ll;
typedef long double ld;
typedef pair<int, int> p32;
typedef pair<ll, ll> p64;
typedef pair<double, double> pdd;
typedef vector<ll> v64;
typedef vector<int> v32;
typedef vector<vector<int>> vv32;
typedef vector<vector<ll>> vv64;
typedef vector<vector<p64>> vvp64;
typedef vector<p64> vp64;
typedef vector<p32> vp32;
```

Example templates

- **Template from GFG :** https://www.geeksforgeeks.org/how-to-setup-competitive-programming-in-visual-studio-code-for-c/ (Does not support inbuilt functions for printing various types of data structures).

How to setup a custom snippet

- **Hard way :**
  - Follow the part to create user snippet in the following blog : https://www.geeksforgeeks.org/how-to-setup-competitive-programming-in-visual-studio-code-for-c/
- **Easy way :**
  - Install this VS Code extension : https://marketplace.visualstudio.com/items?itemName=vscode-snippet.snippet (Extension name : Snippet)
  - Select any piece of code that you want to save, right click and then click on ‘Save Snippet’.