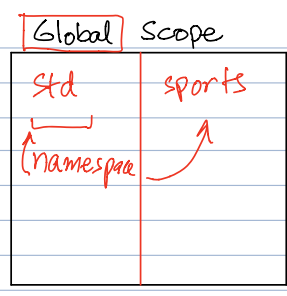


```
#include <iostream>
using namespace std;
```



```
int main () {
    std::cout << "Hello world" << endl;
    return 0;
}
```

std::cout ← console output  
 sports::cout ← captain of —

```
int max (int a, int b) {
    // return the max of the two
}
```

```
template <class T>
T max (T a, T a) {
    // ...
}
```

```
max<int>(5, 6)
```

```
max<string>("Usman", "Ali")
```

function template  
 "generalized version of a function that can be 'specialized' later".

• Identical

```

template <class T>
class List {
    struct node {
        T val;
    };
    void pop();
    void push(int in);
};

```

Diagram illustrating the template instantiation process:

- The `int` in `push(int in);` is circled in green, with a red arrow pointing to the word `student`.
- The `int` in the `node` struct is circled in green, with a red arrow pointing to the word `student`.
- The `int` in the `pop()` method is circled in green, with a red arrow pointing to the word `student`.
- The `int` in the `push` method signature is circled in green, with a red arrow pointing to the word `student`.

```

List<int> l;
List<Student> students;

```

"class template" —  
 generics —