



**Computational Frameworks Recognizer**

**Framework backbone logic stripping & Library framework template instantiation with the custom business logic**

```
#include "Fold.h"
using namespace std;

class Elem : public FoldElem::Element {
public:
    void grow() override { value = 1; }
    int value;
};

int main() {
    int foldDepth = 5;
    FoldElem f(foldDepth);
    Elem e1(1);
    Elem e2(2);
    Elem e3(3);
    Elem e4(4);
    Elem e5(5);
    f.add(e1);
    f.add(e2);
    f.add(e3);
    f.add(e4);
    f.add(e5);
    f.fold();
    int result;
    ComputeFwd comp(f);
    result = fold.comp(f);
    // fold = 15, <= 4, < 5
    // result = 3
    return 0;
}
```

Listing 4.2: Left fold computation using our Fold computational framework



**Regular Parallel Compilation**

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

int main() {
    long long int sum = 0;
    // int = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
    int result = 0;
    for (auto it = 1; it <= 10; ++it) {
        int x = result;
        result = x + it;
    }
    // result = 55
    return 0;
}
```

Listing 4.1: Left fold computation using standard STL list class template

**Original Source Code**