



Optional- Passing Functions

Problem statement

Exercise

The following function somehow accumulates the numbers 1 through n. Write a main program which calls this twice with appropriate functions so that the sum of the numbers and then the product is returned.

```
int accumulate(int n, function<int(int,int)> f){
    int res = 1;
    for(int i=2; i<=n; i++)
        res = f(res,i);
    return res;
} // works only for positive n.
```



Solution- With a normal method

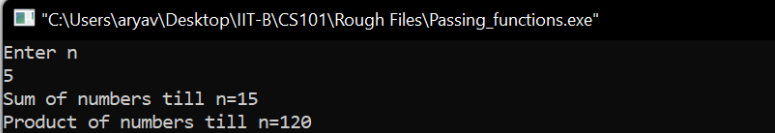
```
1 #include<iostream>
2 #include<functional> //We need this else we can't write std::function<>
3
4 int accumulate(int n, std::function<int(int,int)> f){
5     int res=1;
6     for(int i=2;i<=n;i++)
7         res=f(res,i);
8     return res;
9 }
10
11 int sum(int a, int b){
12     return a+b;
13 }
14
15 int product(int a, int b){
16     return a*b;
17 }
18
19 int main(){
20     std::cout<<"Enter n"<<std::endl;
21     int n;
22     std::cin>>n;
23     std::cout<<"Sum of numbers till n="<<accumulate(n,sum)<<std::endl;
24     std::cout<<"Product of numbers till n="<<accumulate(n,product)<<std::endl;
25     return 0;
26 }
27
```

```
"C:\Users\aryav\Desktop\IIT-B\CS101\Rough Files\Passing_functions
Enter n
5
Sum of numbers till n=15
Product of numbers till n=120

Process returned 0 (0x0)   execution time : 8.972 s
Press any key to continue.
```

With lambda expression

```
1 #include<iostream>
2 #include<functional> //We need this else we can't write std::function<>
3
4 int accumulate(int n, std::function<int(int,int)> f){
5     int res=1;
6     for(int i=2;i<=n;i++)
7         res=f(res,i);
8     return res;
9 }
10
11 int main(){
12     std::cout<<"Enter n"<<std::endl;
13     int n;
14     std::cin>>n;
15     std::cout<<"Sum of numbers till n="<<accumulate(n,[](int a, int b){return a+b;})<<std::endl;
16     std::cout<<"Product of numbers till n="<<accumulate(n,[](int a, int b){return a*b;})<<std::endl;
17     return 0;
18 }
19
```



Basically a lambda function is just a nameless pathetic function that is mildly useful because we don't have to define an entire function for the sake of passing it this way. The syntax is

```
// [](formal_parameters){body with a return}
// e.g. [](int a, int b){int c=a-b; return c*a;} Note how we need not even specify return type of our nameless function
using namespace std;
cout<<"Here is a really weird way to add 3 and 5"<<endl;
cout<<[](int a, int b){return a+b;}(5,3); //This can be read as f(5,3) i.e. we are passing 5 and 3 as parameters to our function

//If you want to force a return type use this
cout<<[](int a, int b)->double{return a+b;}(3,5); //prints 8.0 and not 8
```