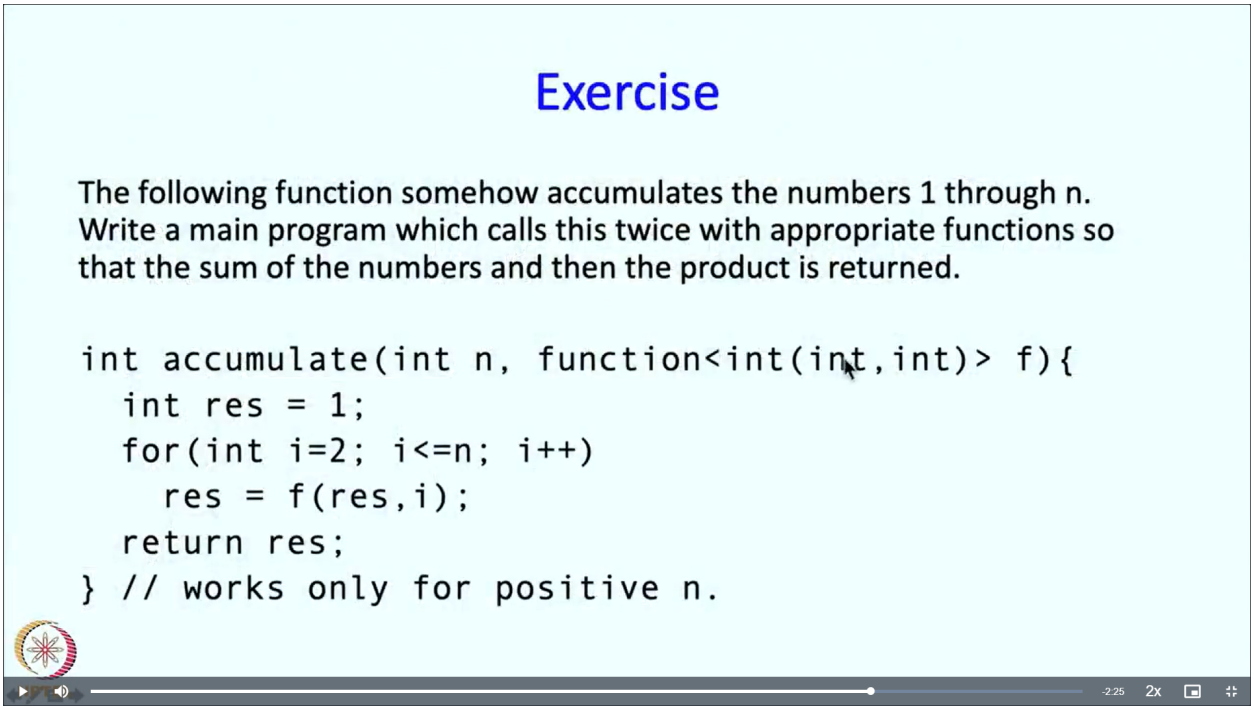
Optional- Passing Functions

Problem statement



Solution- With a normal method

```cpp
#include<iostream>
#include<functional> //We need this else we can't write std::function<>

int accumulate(int n, std::function<int(int,int)> f){
    int res=1;
    for(int i=2;i<=n;i++)
        res=f(res,i);
    return res;
}

int sum(int a, int b){
    return a+b;
}

int product(int a, int b){
    return a*b;
}

int main(){
    std::cout<<"Enter n"<<std::endl;
    int n;
    std::cin>>n;
    std::cout<<"Sum of numbers till n="<<accumulate(n,sum)<<std::endl;
    std::cout<<"Product of numbers till n="<<accumulate(n,product)<<std::endl;
    return 0;
}
```

```
"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

```cpp
#include<iostream>
#include<functional> //We need this else we can't write std::function<>

int accumulate(int n, std::function<int(int,int)> f){
    int res=1;
    for(int i=2;i<=n;i++)
        res=f(res,i);
    return res;
}

int main(){
    std::cout<<"Enter n"<<std::endl;
    int n;
    std::cin>>n;
    std::cout<<"Sum of numbers till n="<<accumulate(n,[](int a, int b){return a+b;})<<std::endl;
    std::cout<<"Product of numbers till n="<<accumulate(n,[](int a, int b){return a*b;})<<std::endl;
    return 0;
}
```

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

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

```cpp
// [](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
```