CS 31 Week 4 Discussion 2E

Srinath

Announcements

- Project 3 is up! Due **11:00 PM, Wednesday, October 26**

- Midterm 1 is on **Tuesday, October 25,** Select your preferable timing by visiting link in the announcements page.

Outline

- Functions
- Project 3 : Tips
- Worksheet 4

Functions

Functions : Definition

A group of statements that perform a defined task.

```
returnType functionName( arg1Type arg1Name, arg2Type arg2Name, … )
{

        // .. the function body

}
```

Example :

```
int square(int x) {
        int result = x*x;
        return result;
}
```

Functions : Definition

Why do we need functions?

A group of statements that perform a defined task.

```
returnType functionName( arg1Type arg1Name, arg2Type arg2Name, … )
{

        // .. the function body

}
```

Example :

```
int square(int x) {
        int result = x*x;
        return result;
}
```

Functions : Definition

Why do we need functions?

- To simplify our work
- Incremental development
- Reuse at various places
- Easy to test and debug.

A group of statements that perform a defined task.

```
returnType functionName( arg1Type arg1Name, arg2Type arg2Name, … )
{

        // .. the function body

}
```

Example :

```
int square(int x) {
        int result = x*x;
        return result;
}
```

Functions : Definition

A group of statements that perform a defined task.

```
returnType functionName( arg1Type arg1Name, arg2Type arg2Name, … )
{

        // .. the function body

}
```

Example :

```
int square(int x) {
        int result = x*x;
        return result;
}
```

Why do we need functions?

- To simplify our work
- Incremental development
- Reuse at various places
- Easy to test and debug.

Do all functions need to return some value?

Functions : Definition

A group of statements that perform a defined task.

```
returnType functionName( arg1Type arg1Name, arg2Type arg2Name, … )
{

        // .. the function body

}
```

Example :

```
int square(int x) {
        int result = x*x;
        return result;
}
```

Why do we need functions?

- To simplify our work
- Incremental development
- Reuse at various places
- Easy to test and debug.

Do all functions need to return some value?

- Not necessarily
- We have **void** return type so that functions don't need to return anything.

Example :

```
void printRating(int n) {
        for(int i=0; i<n; i++){
                cout<< "*";
        }
        cout<<endl;
}
```

Functions : Definition

A group of statements that perform a defined task.

```
returnType functionName( arg1Type arg1Name, arg2Type arg2Name, … )
{

        // .. the function body

}
```

Example :

```
int square(int x) {
        int result = x*x;
        return result;
}
```

Why do we need functions?

- To simplify our work
- Incremental development
- Reuse at various places
- Easy to test and debug.

Do all functions need to return some value?

- Not necessarily
- We have **void** return type so that functions don't need to return anything.

Example :

```
void printRating(int n) {
        for(int i=0; i<n; i++){
                cout<< "*";
        }
        cout<<endl;
}
```

Is there any limit on maximum number arguments we can define?

Functions : Definition

A group of statements that perform a defined task.

```
returnType functionName( arg1Type arg1Name, arg2Type arg2Name, … )
{

        // .. the function body

}
```

Example :

```
int square(int x) {
        int result = x*x;
        return result;
}
```

Why do we need functions?

- To simplify our work
- Incremental development
- Reuse at various places
- Easy to test and debug.

Do all functions need to return some value?

- Not necessarily
- We have **void** return type so that functions don't need to return anything.

Example :

```
void printRating(int n) {
        for(int i=0; i<n; i++){
                cout<< "*";
        }
        cout<<endl;
}
```

Is there any limit on maximum number arguments we can define?

- No

Functions : Calling

```
functionName(arg1, arg2, ...)
```

Using the returned value

```
returnType var1 = functionName(arg1, arg2, ...);
```

Functions : Calling

functionName(arg1, arg2, ...)

Using the returned value

returnType var1 = functionName(arg1, arg2, ...);

```
int square(int x) {
        int result = x*x;
        return result;
}

int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
}
```

Functions : Examples

```
int cube(int x) {
        int result = x*(x*x);
}
```

Will this compile?

Functions : Examples

```
int cube(int x) {
        int result = x*(x*x);
}
```

Will this compile? - No

```
string cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Will this compile?

Functions : Examples

```
int cube(int x) {
        int result = x*(x*x);
}
```

Will this compile? - No

```
string cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Will this compile? - No

```
int cube(int x) {
        int result = x*(x*x);
        return -1;
}
```

Will this compile?

Functions : Examples

```
int cube(int x) {
        int result = x*(x*x);
}
```

Will this compile? - No

```
string cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Will this compile? - No

```
int cube(int x) {
        int result = x*(x*x);
        return -1;
}
```

Will this compile? - Yes
Does it do what we need?

Functions : Examples

```
int cube(int x) {
        int result = x*(x*x);
}
```

Will this compile? - No

```
string cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Will this compile? - No

```
int cube(int x) {
        int result = x*(x*x);
        return -1;
}
```

Will this compile? - Yes
Does it do what we need? - No

```
int cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Compiles and does what we need.

Functions : Pre-defined

C++ has libraries with defined functions for you.

Functions : Pre-defined

C++ has libraries with defined functions for you.

How do you calculate square root of a given number?

Functions : Pre-defined

C++ has libraries with defined functions for you.

How do you calculate square root of a given number?

Don’t worry, someone has already written a function to do that.
You just have to make use of it.

```
#include <cmath>

double sqrt(double x);
```

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

int main() {
        cout<< “square root of 2 is ”<<sqrt(2)<<endl;
        return 0;
}
```

Functions : Pre-defined

C++ has libraries with defined functions for you.

How do you calculate square root of a given number?

Don’t worry, someone has already written a function to do that.
You just have to make use of it.

```
#include <cmath>

double sqrt(double x);
```

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

int main() {
	cout<< “square root of 2 is ”<<sqrt(2)<<endl;
	return 0;
}
```

But,.... How do you actually calculate square root of a given number?

Functions : Pre-defined

C++ has libraries with defined functions for you.

How do you calculate square root of a given number?

Don’t worry, someone has already written a function to do that.
You just have to make use of it.

```
#include <cmath>

double sqrt(double x);
```

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

int main() {
        cout<< “square root of 2 is ”<<sqrt(2)<<endl;
        return 0;
}
```

But,.... How do you actually calculate square root of a given number?

- Check out Newton-Raphson method.

Functions : Pre-defined

C++ has libraries with defined functions for you.

Some other examples?

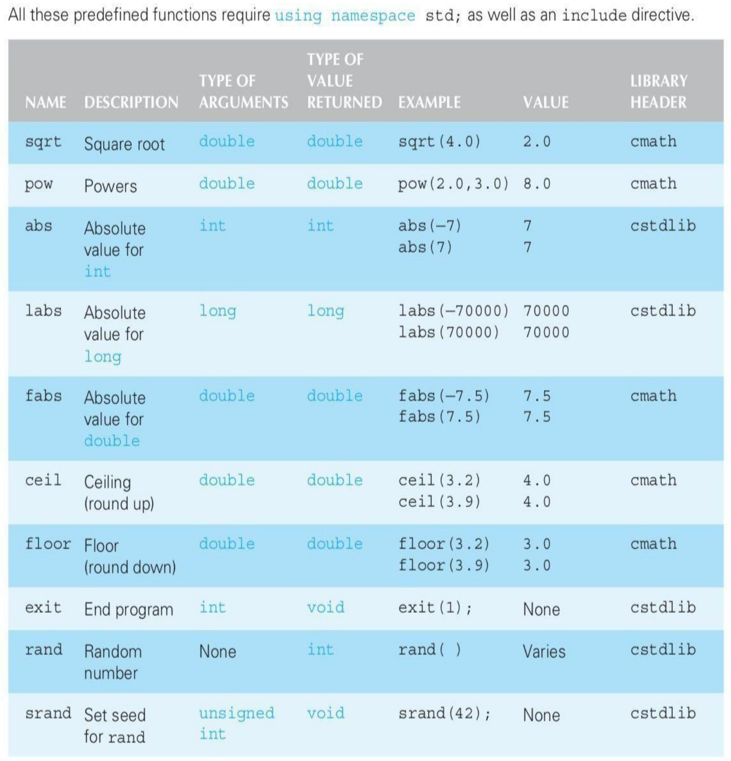
Functions : Pre-defined

C++ has libraries with defined functions for you.

Some other examples?

isdigit()
isalpha()
isupper()
islower()
Above are included with header <cctype>

size()
substr(..)
etc..



All these predefined functions require `using namespace std;` as well as an `include` directive.

NAME	DESCRIPTION	TYPE OF ARGUMENTS	TYPE OF VALUE RETURNED	EXAMPLE	VALUE	LIBRARY HEADER
`sqrt`	Square root	`double`	`double`	`sqrt(4.0)`	`2.0`	`cmath`
`pow`	Powers	`double`	`double`	`pow(2.0,3.0)`	`8.0`	`cmath`

`abs`	Absolute value for `int`	`int`	`int`	`abs(-7)`
				`abs(7)`
					`7`
					`7`
						`cstdlib`

`labs`	Absolute value for `long`	`long`	`long`	`labs(-70000)`
				`labs(70000)`
					`70000`
					`70000`
						`cstdlib`

`fabs`	Absolute value for `double`	`double`	`double`	`fabs(-7.5)`
				`fabs(7.5)`
					`7.5`
					`7.5`
						`cmath`

`ceil`	Ceiling (round up)	`double`	`double`	`ceil(3.2)`
				`ceil(3.9)`
					`4.0`
					`4.0`
						`cmath`

`floor`	Floor (round down)	`double`	`double`	`floor(3.2)`
				`floor(3.9)`
					`3.0`
					`3.0`
						`cmath`

`exit`	End program	`int`	`void`	`exit(1);`	None	`cstdlib`
`rand`	Random number	None	`int`	`rand( )`	Varies	`cstdlib`
`srand`	Set seed for `rand`	`unsigned int`	`void`	`srand(42);`	None	`cstdlib`

Functions :

```
int square(int x) {
        int result = x*x;
        return result;
}

int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}
```

Will this compile?

Functions :

```
int square(int x) {
        int result = x*x;
        return result;
}

int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}
```

Will this compile?

- Yes

Functions :

```
int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}

int square(int x) {
        int result = x*x;
        return result;
}
```

Will this compile?

Functions :

```
int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}

int square(int x) {
        int result = x*x;
        return result;
}
```

Will this compile?

- No, it doesn’t know what square(..) is

Functions : Declaration

```
int square(int x); // declaration

int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}

int square(int x) {
        int result = x*x;
        return result;
}
```

Will this compile?

- No, it doesn’t know what square(..) is

For this to compile, you have to **declare** the function before using it.

C++ provides a way to declare function and later write its implementation.

returnType functionName(arg1Type arg1Name, arg2Type arg2Name, …) **;**

Functions : Pass by Value

When you pass the argument to call a function, Just the value of the **argument is copied** and the argument is **not modified** in the calling function.

```
void printRating(int n) {
        while(n>0){
                cout<< "*";
                n = n-1;
        }
        cout<<endl;
        return;
}

int main() {
        int k = 5;
        cout<< "Value of k before : " << k <<endl;
        printRating(k);
        cout<< "Value of k after : " << k <<endl;
        return 0;
}
```

Output:

```
        Value of k before : 5
        *****
        Value of k after : 5
```

Functions : Pass by Reference

When you pass the argument to call a function, the argument is **modified** in the calling function (if it is modified in your function).

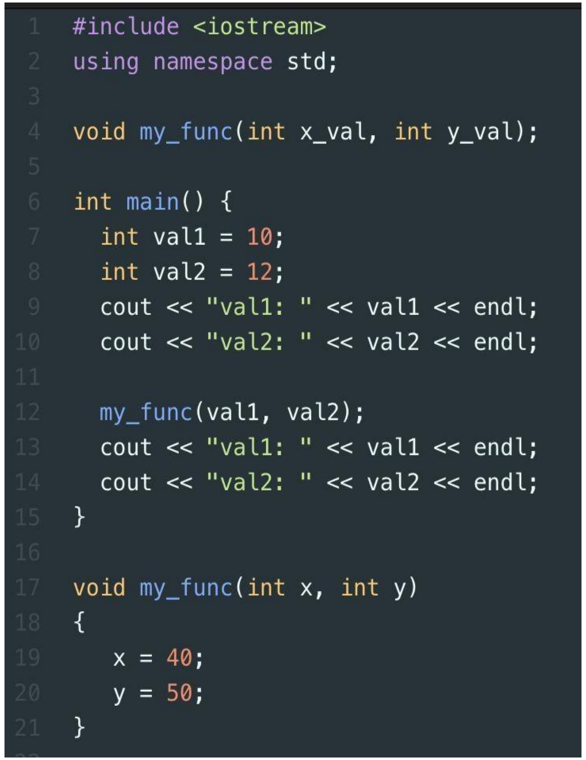
i,e, when your function takes an argument using **& (reference),** it is just another name for the same variable.

```
void printRating(int& n) {
        while(n>0){
                cout<< "*";
                n = n-1;
        }
        cout<<endl;
        return;
}

int main() {
        int k = 5;
        cout<< "Value of k before : " << k <<endl;
        printRating(k);
        cout<< "Value of k after : " << k <<endl;
        return 0;
}
```

Output:

```
Value of k before : 5
*****
Value of k after : 0
```

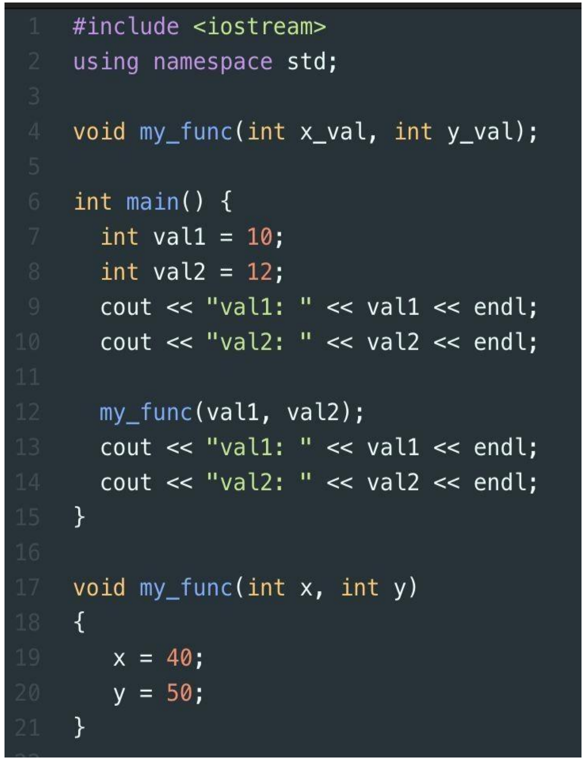
Functions : Practice



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

void my_func(int x_val, int y_val);

int main() {
  int val1 = 10;
  int val2 = 12;
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;

  my_func(val1, val2);
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;
}

void my_func(int x, int y)
{
   x = 40;
   y = 50;
}
```

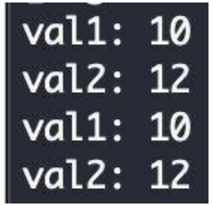
Output ??

Functions : Practice



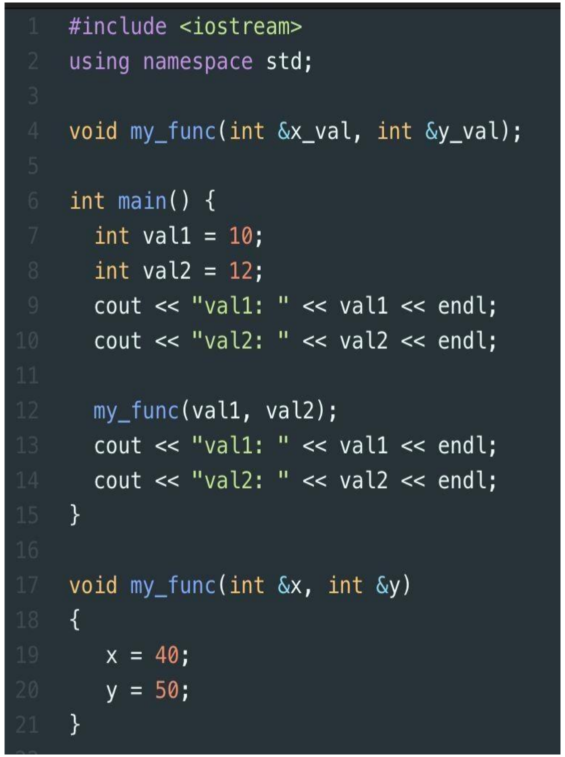
```
#include <iostream>
using namespace std;

void my_func(int x_val, int y_val);

int main() {
  int val1 = 10;
  int val2 = 12;
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;

  my_func(val1, val2);
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;
}

void my_func(int x, int y)
{
   x = 40;
   y = 50;
}
```

Output ??



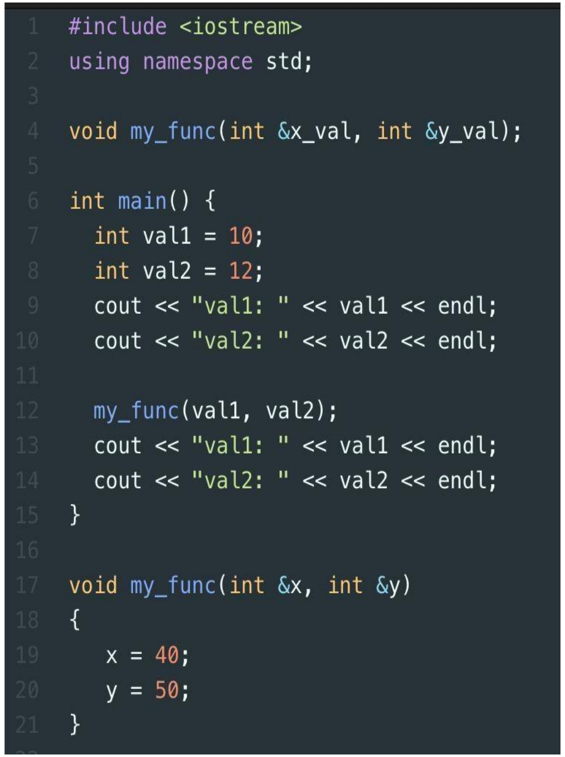
```
val1: 10
val2: 12
val1: 10
val2: 12
```

Functions : Practice


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

void my_func(int &x_val, int &y_val);

int main() {
  int val1 = 10;
  int val2 = 12;
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;

  my_func(val1, val2);
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;
}

void my_func(int &x, int &y)
{
   x = 40;
   y = 50;
}
```

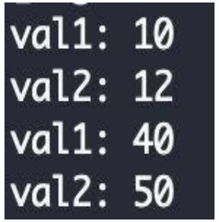
Output ??

Functions : Practice



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

void my_func(int &x_val, int &y_val);

int main() {
  int val1 = 10;
  int val2 = 12;
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;

  my_func(val1, val2);
  cout << "val1: " << val1 << endl;
  cout << "val2: " << val2 << endl;
}

void my_func(int &x, int &y)
{
   x = 40;
   y = 50;
}
```

Output ??



```
val1: 10
val2: 12
val1: 40
val2: 50
```

Functions : Practice

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

// function declaration
void swap(int x, int
y);

int main () {
   // local variable declaration:
   int a =
   100;  int b
   = 200;

   cout << "Before swap, val of a :" << a <<
   endl; cout << "Before swap, val of b :" << b
   << endl;
   swap(a,b);
   cout << "After swap, val of a :" << a <<
   endl; cout << "After swap, val of b :" << b
   << endl;

   return 0;
}
```

```
// swap the values of two numbers
void swap(int x, int y) {
    int temp;
    temp = x; /* save the value of x */
    x = y; /* put y into x */
    y = temp; /* put x into y */
    return;
}
```

Output ??

Functions : Practice

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

// function declaration
void swap(int x, int
y);

int main () {
   // local variable declaration:
   int a =
   100;  int b
   = 200;

   cout << "Before swap, val of a :" << a <<
   endl; cout << "Before swap, val of b :" << b
   << endl;
   swap(a,b);
   cout << "After swap, val of a :" << a <<
   endl; cout << "After swap, val of b :" << b
   << endl;

   return 0;
}
```

```
// swap the values of two
numbers  void swap(int x, int y)
{
    int temp;
    temp = x; /* save the value of x */
    x = y; /* put y into x */
    y = temp; /* put x into y
    */  return;
}
```

Output ??

```
Before swap, val of a :100
Before swap, val of b :200
After swap, val of a :100
After swap, val of b :200
```

Functions : Practice

```
#include <iostream>
using namespace
std;
// function declaration
void swap(int &x, int &y);

int main () {
   // local variable
   declaration:  int a = 100;
   int b = 200;
   cout << "Before swap, val of a :" << a <<
   endl;  cout << "Before swap, val of b :" << b
   << endl;

   swap(a,b);
   cout << "After swap, val of a :" << a <<
   endl;  cout << "After swap, val of b :" << b
   << endl;
   return
}  0;
```

```
// swap the values of two
numbers          &x, int &y) {
void
    swap(int
    int temp; /* save the value of x
    temp = x;  */
    x = y;      /* put y into x */
    y        = /* put x into y */
    temp;
    return;
}
```

Output ??

Functions : Practice

```
#include <iostream>
using namespace
std;
// function declaration
void swap(int &x, int &y);

int main () {
   // local variable
   declaration:  int a = 100;
   int b = 200;
   cout << "Before swap, val of a :" << a <<
   endl;  cout << "Before swap, val of b :" << b
   << endl;

   swap(a,b);
   cout << "After swap, val of a :" << a <<
   endl;  cout << "After swap, val of b :" << b
   << endl;
   return
}  0;
```

```
// swap the values of two
numbers
void              &x, int &y) {
    swap(int
    int temp; /* save the value of x
    temp = x;  */
    x = y;      /* put y into x */
    y        = /* put x into y */
    temp;
    return;
}
```

Output ??

```
Before swap, val of a :100
Before swap, val of b :200
After swap, val of a :200
After swap, val of b :100
```

Functions : Practice

```
void addRatingAfterName(string &name, int n) {
        while(n>0){
                name = name+“*”;
                n = n-1;
        }
        return;
}

int main() {
        int k = 5;
        string my_name = “Newton”;
        cout<< “Value of k before : ” << k <<endl;
        cout<< “Value of my_name before : ” << my_name <<endl;
        addRatingAfterName(my_name, k);
        cout<< “Value of k after : ” << k <<endl;
        cout<< “Value of my_name after : ” << my_name <<endl;
        return 0;
}
```

Will this compile?

Functions : Practice

```
void addRatingAfterName(string &name, int n) {
        while(n>0){
                name = name+"*";
                n = n-1;
        }
        return;
}

int main() {
        int k = 5;
        string my_name = "Newton";
        cout<< "Value of k before : " << k <<endl;
        cout<< "Value of my_name before : " << my_name <<endl;
        addRatingAfterName(my_name, k);
        cout<< "Value of k after : " << k <<endl;
        cout<< "Value of my_name after : " << my_name <<endl;
        return 0;
}

 }
```

Will this compile?

- Yes

What is the output?

Functions : Practice

```
void addRatingAfterName(string &name, int n) {
        while(n>0){
                name = name+“*”;
                n = n-1;
        }
        cout<<endl;
        return;
}

int main() {
        int k = 5;
        string my_name = “Newton”;
        cout<< “Value of k before : ” << k <<endl;
        cout<< “Value of my_name before : ” << my_name <<endl;
        addRatingAfterName(my_name, k);
        cout<< “Value of k after : ” << k <<endl;
        cout<< “Value of my_name after : ” << my_name <<endl;
        return 0;
}
 }
```

Will this compile?

- Yes

What is the output?

```
Value of k before : 5
Value of my_name before : Newton
Value of k after : 5
Value of my_name after : Newton*****
```

Functions : Practice

What is the output now?

```
void addRatingAfterName(string name, int n) {
        while(n>0){
                name = name+“*”;
                n = n-1;
        }
        return;
}

int main() {
        int k = 5;
        string my_name = “Newton”;
        cout<< “Value of k before : ” << k <<endl;
        cout<< “Value of my_name before : ” << my_name <<endl;
        addRatingAfterName(my_name, k);
        cout<< “Value of k after : ” << k <<endl;
        cout<< “Value of my_name after : ” << my_name <<endl;
        return 0;
}

 }
```

Functions : Practice

```
void addRatingAfterName(string name, int n) {
        while(n>0){
                name = name+“*”;
                n = n-1;
        }
        return;
}

int main() {
        int k = 5;
        string my_name = “Newton”;
        cout<< “Value of k before : ” << k <<endl;
        cout<< “Value of my_name before : ” << my_name <<endl;
        addRatingAfterName(my_name, k);
        cout<< “Value of k after : ” << k <<endl;
        cout<< “Value of my_name after : ” << my_name <<endl;
        return 0;
}

 }
```

What is the output now?

```
Value of k before : 5
Value of my_name before : Newton
Value of k after : 5
Value of my_name after : Newton
```

Functions : Summary

- Declaring a function(Just the declaration)
- Defining a function(implementation)
- Return types, argument types, return values
- Pass by value
- Pass by reference

}

Project 3

Project 3 : Tips

- Start Early!!!

- Focus on incrementally solving the problem.

- Use functions to break down and solve simpler problems

- Make sure to check the writeups in the announcement dated 10/15/22: functions, a technique for processing strings, and a note about characters and integers.

- You are free to use the given code(in the spec) to check for a valid state code.

- Don't try to solve all of a big problem all at once. Instead make assumptions about smaller chunks and solve them in separate functions. Use these separate functions as helpers to solve the big problem.

- Reach out to TA's/LA's in case you need help!

Questions??