Video No.: 71

Topic Name: Static and Dynamic Scoping - Part 3

In dynamic scoping, definition of a variable is resolved by searching its containing block and if not found then searching its calling function and if still not found then the function which called that function will be searched and so on.

Static Scoping Example:

```
#include<stdio.h>
int fun1(int);
int fun2(int);
int a = 5;

int main(){
   int a = 10;
   a = fun1(a);
   printf("a = %d",a);
}

int fun1(int b){
   b = b+10;
   b = fun2(b);
   return b;
}

int fun2(int b){
   int c;
   c = a+b;
   return c;
}
```

For Static Scoping
Output will be

30

Homework: What will be the output?

```
#include<stdio.h>
int a, b;
 void print(){
   printf("%d %d",a,b);
 }

 int fun1(){
   int a,c;
   a = 0;
   b = 1;
   c = 2;
   return c;
 }

 void fun2(){
   int b;
   a = 3;
   b = 4;
   print();
 }

int main(){
   a = fun1();
   fun2();
}
```

Explanation:

// First globally:

// Global a = 0, Global b = 0 taken by the compiler

// Second it enters main( ):

// global a = output generated by fun1( ), global b = 0

// Third it enters fun1( ):

// global a = 0, global b = 1, local a = 0, local c = 2 (as it is now dealing with local variables)

// meanwhile in main( ):

// global a =2

// fourth it enters fun2( ):

// global a = 3, global b = 1, local b = 4 (AS ONLY "b" IS DEFINED LOCALLY)

// Finally print( ):

// Static prints global 'a' and 'b'. (a =3, b=1)

// Dynamic prints the values it finds in immediate function from where print( ) is called (a=3, b =4)