1. Scope and Lifetime of C variables

(a)

A non-static local variable has the scope of the function or block in which the variable is declared. The lifetime of a non-static local variable is the function or block since it is destroyed when the function terminates or the block finishes execution.

(b)

The lifetime of a static local variable is the execution of the entire program; the storage for a static local variable is allocated at the beginning of program execution. However, a static local variable is invisible outside the function where the variable is declared. A static local variable has the scope of the function or block in which the variable is declared similar to (a).

(c)

The scope of a non-static global variable is the entire C program. By using the `extern` keyword, it is visible even outside of the file in which the variable is declared. If a local variable has the same name as the global variable, the global variable will be invisible in the scope of the local variable. The lifetime of a non-static global variable is the entire program execution.

(d)

The scope of a static global variable is the scope of the file in which the variable is declared. It is invisible outside the file it is declared. Similar to (c), if a local variable has the same name as the global variable, the global variable will be invisible in the scope of the local variable. The lifetime of a static global variable is the entire program execution.

2. PHP namespace mechanism

Namespaces in PHP solve the possible naming collision of classes, functions, and variables. Using namespace guarantee that identifiers are unique in case of name collisions.

```php
<?php
namespace mynamespace;
class myClass {
  public function func () {echo "Hello World";}
}
?>
```

```php
<?php
namespace secondnamespace;
class myClass {
      public function func () {echo "Goodbye World";}
}
$obj = new \mynamespace\myClass(); // Using the myClass of the previous namespace
$obj->func(); // accessing methods of the object
?>
```

Without the namespace there would be a name collision between two myClass classes and the output would be “Goodbye World” instead of “Hello World”.

Source: https://www.php.net/manual/en/language.namespaces.php

3. Ada Dynamic and Static Scoping

(a) Assuming that static scoping is used, say which variables are visible in the bodies of each of the procedures: Main, Sub1, Sub2 and Sub3.

Answer:

i. Main: <A, 2> <B, 2> <C, 2>
ii. Sub1: <D, 4> <E, 4> <A, 2> <B, 2> <C, 2>
iii. Sub2: <D, 9> <C, 9> <A, 2> <B, 2>
iv. Sub3: <B, 11> <D, 11> <F, 11> <C, 9> <A, 2>

(b) Assuming that dynamic scoping is used and the calling sequence is Main calls Sub1; Sub1 calls Sub2; Sub2 calls Sub3, say which variables are visible in Sub3.

Answer:

i. From Sub3: <B, 11> <D, 11> <F, 11>
ii. From Sub2: <C, 9>
iii. From Sub1: <E, 4>
iv. From Main: <A, 2>

(c) Assuming that dynamic scoping is used and the calling sequence is Main calls Sub2; Sub2 calls Sub3; Sub3 calls Sub1, say which variables are visible in Sub1.

Answer:

i. From Sub1: <D, 4> <E, 4>
ii. From Sub3: <B, 11> <F, 11>
iii. From Sub2: <C, 9>
iv. From Main: <A, 2>

(d) Assuming that dynamic scoping is used and the calling sequence is Main calls Sub2; Sub2 calls Sub1, say which variables are visible in Sub1.

Answer:

i. From Sub1: <D, 4> <E, 4>
ii. From Sub2: <C, 9>
iii. From Main: <A, 2>, <B, 2>