1. Provide code and necessary elaborations for demonstrating how you use dynamic scoping in Task 2.

I use deep access method of dynamic scoping. Which I can store and access the variable in other subprograms.

For example:

There is a sub program in Server.pm

```
sub task_attr {
        my $self = shift;
        my $task = shift @_;
        return $task->name(), $task->pid(), $task->time();
}
```

When this subprogram is called, it will go to the subprograms of the input task object->name(),pid(),and time()

Main() calls task_attr()

task_attr calls name() , pid() , time()

time	Local: time	
	Dynamic links to task_attr	
	Return to task_attr	

pid	Local: pid	
	Dynamic links to task_attr	
	Return to task_attr	

name	Local: name	
	Dynamic links to task_attr	
	Return to task_attr	

task_attr	Dynamic link to main	
	Return (to main)	

Main	X	->name
	Y	->pid
	Z	->time

2. What's the advantage and disadvantage of using dynamic scoping, compared with lexical scoping?

Advantages of dynamic scoping:

- Writability. It is more convenient, and it is more flexible than lexical scoping. For example, in some cases, some parameters passed from one subprogram to another are variables that are defined in the caller. None of these need to be passed in a dynamic scoped language.

Disadvantages of dynamic scoping:

- Highest possibility of causing bugs, since it's Inability to statically check for references to nonlocal variables; and can't protect local variables from being accessed by subprograms because local variables of subprogram are all visible to all other executing subprograms.
- Decrease in readability, because the calling sequence of subprograms must be known to determine the meaning of references to nonlocal variables.
- Slower execution speeds, because every activation record instance in the chain must be searched until the first of it is found.
- Need more memory space, because the activation records need to store the names of variables for the search process. When lexical scoping only need to store the values.