第 11 讲书面作业包括两部分。第一部分为 Lecture11.pdf 中课后作业题目中的 1、3、4 和 5 题。第二部分为以下题目：

A1.

以下是某简单语言的一段代码。语言中不包含数据类型的声明，所有变量的类型默认为整型（假设占用一个存储单元）。语句块的括号为‘begin’和‘end’组合；赋值号为 ‘:=’。每一个过程声明对应一个静态作用域。该语言支持嵌套的过程声明，但只能定义无参过程，且没有返回值。过程活动记录中的控制信息包括静态链 SL，动态链 DL，以及返回地址 RA。程序的执行遵循静态作用域规则。

```
（1） var a0,b0;
（2） procedure fun1 ;
（3）         var a1,b1;
（4）         procedure fun2 ;
（5）                 var a2;
（6）                 begin
（7）                         a2:= a1*a0-b1;
（8）                         if(a2<0) then call fun3;
   .                          ……    /*不含任何 call 语句和声明语句*/
   .                  end;
   .          begin
                      a1:= a0 - b0;
                      b1:=a0 + b0;
（x）                 If   a1 < a0   then   call fun2 ;
   .                  ……      /*不含任何 call 语句和声明语句*/
   .          end ;
   . procedure fun3 ;
   .          var a3,b3;
   .          begin
                      a3:=a0*b0 ;
                      b3:=a0/b0 ;
（y）                 if(a3 <> b3) call fun1 ;
 .                    ……    /*不含任何 call 语句和声明语句*/
   .          end ;
   . begin
   .          a0 := 1;
   .          b0 := 2;
   .          call fun3;
   .          ……    /*不含任何 call 语句和声明语句*/
   . end .
```

试问：当过程fun1被第二次激活时，运行栈上共有几个活动记录？分别是那些过程的活动记录？当前位于栈顶和次栈顶的活动记录中静态链 SL 和动态链 DL 分别指向什么位置？（注：指出是哪个活动记录的起始位置即可）

参考解答：

当过程fun1被第二次激活时，运行栈上共有6个活动记录；包括 main , fun3,fun1,fun2,fun3, fun1。 当前位于栈顶的活动记录的静态链 SL指向运行栈的起始位置，即主过程的活动纪录起始位置；动态链 DL指向过程fun3第二个实例所对应的活动记录的起始位置。当前位于次栈顶的活动记录的静态链 SL指向运行栈的起始位置，即主过程的活动纪录起始位置；动态链 DL指向过程fun2第一个实例所对应的活动记录的起始位置。

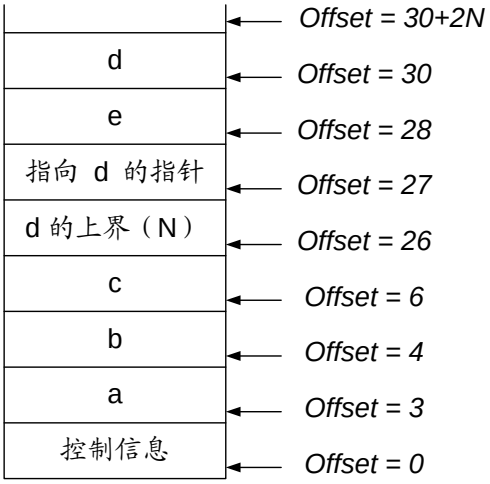
以下是 Lecture11 文档中的题目

1.

若按照某种运行时组织方式，如下函数 p 被激活时的过程活动记录如图 2 所示。其中 d 是动态数组。

```
static int N;

void p( int a)   {
    float b;
    float c[10];
    float d[N];
    float e;
    …
}
```



试指出函数 p 中访问 d[i]（$0 \le i < N$）时相对于活动记录基址的 *Offset* 值如何计算？若将数组 c 和 d 的声明次序颠倒，则d[i]（$0 \le i < N$）又如何计算？（对于后一问题默认采用同样的运行时组织方式，若你认为可能有歧义，请予以说明）

参考解答：

函数 p 中访问 d[i]（$0 \le i < N$）时相对于活动记录基址的 *Offset* 值可通过 30+2i 来计算。

若将数组 c 和 d 的声明次序颠倒，d[i]的计算方式不变。

3. 若在第2题中，我们采用 Display 表来代替静态链。假设采用只在活动记录保存一个Display 表项的方法，且该表项占居图中SL的位置。（1）指出当前运行状态下

Display 表的内容；（2）指出各活动记录中所保存的Display 表项的内容（即图中所有SL位置的新内容）

参考解答：

(1) 当前 Display 表的内容

D[0] = 0

D[1] = 22

D[2] = 13

(2) 各活动记录中所保存的 Display 表项的内容

单元0中的内容： _ 无效

单元5中的内容： _ 无效

单元9中的内容：5

单元13中的内容： _ 无效

单元18中的内容： 9

单元22中的内容：18

4. 若采取动态作用域规则，该程序的执行效果与之前有何不同？

参考解答：在第二次执行到语句 L 时，若是静态作用域规则，则 a=1, b=2，因此会再次调用 P; 若是动态作用域规则，则 a=3, b=2，因此不会调用 P。

5. 对于下图中的 Decaf/Mind 程序，（1）根据课程实验所采取的运行时存储组织方式，当变量 a 所指向的对象创建后，其对象存储空间中依次存放哪些内容？（2）class Apple 的 vtable 中依次存放哪些内容？

```
class Fruit
{
    int price;
    string name;
    void init(int p, string s)  {price=p; name=s;}
    void print(){ Print(” The price of ", name, " is ",price,"\n");}
}
class Apple extends Fruit
{
    string color;
    void setcolor(string c)  {color=c;}
    void print(){
        Print(“The price of ",color," ",name," is ", price,"\n");
    }
}
class Main {
{
    class Apple a;
    a=New Apple();
    a.setcolor("red");
    a.init(100,"apple");
    a.print();
}
```

参考解答：

（1）a 所指向的对象存储空间中依次存放：指向 Class Apple vtable 的指针，int 变量 price，name 和 color.

（2）class Apple 的 vtable 中包含内容依次为：指向 class Fruit 的 vtable 的指针，指向 class Apple 类名字串的指针，class Fruit 的 init 函数代码入口指针， class Apple 的 print 函数代码人口的指针，以及 class Apple 的 setcolor 函数代码人口的指针。