Imperative Programming

T2.1:

What does the program print if the language uses dynamic scope? Why?

y>0:

print: 1

y<0:

print: 0

because when y >0, the function write () executed, the association for x is sought in the same block, which has been definited as 1, in line 2. but in the function f1(), which has been called in y>0 block, there exists a sentence (x = 0;). according to the dynamic scope, the x in function f1() has been definated in the block , which is int x=4, and x has been destoried in function f2(), int x =5. So x was equal to 4 and changed to 0. so the ouput should be 1, because the x=1 doesn't change.

When y<0, the function write() executed, the association for x is sought in the same block, which has been definited as 1, in line 2. accoding to previous steps, the x in function f1() can not find a valid association in y<0 block, then it will find the x has been definited as 1, in line 2. so x =1 has been change to 0. the output will be 0.

T2.2:

What does the program print if the language uses static scope and passes parameters by reference. Why?

Print:

5

7

the first write print the value 5, while the second one prints the value 7.

This is because: The procedure f1() is declared in the same block . Thus, int x=5 and f1(x) effectively associates x to 7 in first block.

In the case of the fist call of the write, we look if there exists a valid association for x in such nested block, it does exist and it is 5.

In the case of the second call of the write, we look if there exists a valid association for x in such nested block, it does exist and it is the first definiation for x, and in Function f1(), the

value of x has been changed due to the static scope. So x = x+y is executed, y=2 based on the passing paramenter by refereence, so x=7.

T2.3:

What does the program print if the language uses static scope and passes parameters by value. Why?

Print:

6

7

in the case of the frist call of the function write(), we look if there exists a valid association for x in such nested block, it does exist and it is 6 due to x++, which will increment it by one.

In the case of the second call of the function write(), we look if there exists a valid association for x infind . -type f -exec du -a {} + | sort -n -r | less such nested block, it exists and it is the first defination of x. but in function f1(x++), the parameter x is equal to 5 after calling function increments it by one. In other word, firstly, the programming call function f1(x) and then execution x=x+1. So we can get the frist defination x is x= x+y => x= 2+5 => x=7

T2.4

What does the program print if the language uses static scope and passes parameters by name. Why?

Print:

5

7

In the case of the first call of the function write(), we look if there exists a valid association for x in such nested block, it does exist and it is 5 due to f1(x++) passing parameter by name. In call-by-name, the actual parameter is passed unevaluated, and the address is worked out each time the formal parameter is used. In other words, in function f1(x++),

x = x+y => x = x + y++ => x = x+y; y = y+1;

the result will be x = 5 and y =6, but the function f1() doesn't return any variables. So the ouput will be 5

in the case of the first call of the function write(), we look if there exists a valid association for x in such nested block, it does exist and has been change by the formula x = x +y, so the result will be 7.