605.629: Programming Languages
Assignment 4
Sabbir Ahmed

October 2, 2021

1. [20 pts, scope]

Consider the following Ada skeletal program:

```
procedure Main is
    X : Integer;
    procedure Sub1 is
        begin -- of Sub1
        Put(X);
        end; -- of Sub1
    procedure Sub2 is
        X : Integer;
        begin -- of Sub2
            X := 10;
            Sub1;
        end; -- of Sub2
        begin -- of Main
            X := 5;
            Sub2;
        end; -- of Main
```

a. What value of X is printed in procedure Sub1, when static scoping used?

Answer:

5

---

b. What value of X is printed in procedure Sub1, when dynamic scoping used?

Answer:

10

---

2. [30 pts, scope]

Consider the following program:

```
void fun1(void); /* prototype */
```

```
void fun2(void); /* prototype */
void fun3(void); /* prototype */
void main() {
    int a, b, c;
    ...
}
void fun1(void) {
    int b, c, d;
    ...
}
void fun2(void) {
    int c, d, e;
    ...
}
void fun3(void) {
    int d, e, f;
    ...
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Provide the name of the function that the variable is defined.

a. Main call fun1, fun1 calls fun2, fun2 calls fun3

Answer:

a (defined in main)
b (defined in fun1)
c (defined in fun2)
d, e, f (defined in fun3)

---

b. Main call fun1, fun1 calls fun3

Answer:

a (defined in main)
b, c (defined in fun1)
d, e, f (defined in fun3)

---

c. Main calls fun2, fun2 calls fun3, fun3 calls fun1

Answer:

a (defined in main)
e, f (defined in fun3)
b, c, d (defined in fun1)

---

d. Main calls fun3, fun3 calls fun1

Answer:

a (defined in main)
e, f (defined in fun3)
b, c, d (defined in fun1)

---

e. Main calls fun1, fun1 calls fun3, fun3 calls fun2

Answer:

a (defined in main)
b (defined in fun1)
f (defined in fun3)
c, d, e (defined in fun2)

---

f. Main calls fun3, fun3 calls fun2, fun2 calls fun1

Answer:

a (defined in main)
f (defined in fun3)
e (defined in fun2)
b, c, d (defined in fun1)

---

3. [5 pts, L-value, r-value]

Consider the following C statements. Comment about their validity and the l-value, r-value values of the variables. (Hint: You might try a C compiler if necessary)

a. int var = 10;

Answer:

Valid syntax
The lvalue is `var` which is being declared as an int. The rvalue is the integer 10.

---

b. int* good_addr = &(var + 1);

Answer:

Compilation error
The lvalue is `good_addr` and the rvalue is `var + 1`, where `var` was declared an int. The unary `&` (address-of) operator requires an lvalue as its operand. If the expression was modified to `&var + 1` then it would be a valid expression.

---

c. int* addr = &var;

Answer:

Valid syntax
The lvalue is `addr` and the rvalue is `var`. This expression is assigning the address of `var` to the pointer `addr`.

---

d. &var = 20;

Answer:

Compilation error
The rvalue is `&var`. The unary `&` (address-of) operator requires an lvalue as its operand.

---

e. *++var = 30;

Answer:

Compilation error
`var` was declared as an int and not a pointer. The left hand expression `*++var` cannot be computed and dereferenced.

---

f. ++*var = 40;

Answer:

Compilation error
`var` was declared as an int and not a pointer. The left hand expression `++*var` cannot be computed and dereferenced.

---

g. int* var = (void *)50;

Answer:

Valid syntax if this expression is independent from the above portions, where `var` had already been declared an int. The lvalue is `var` being declared as a pointer and being assigned the pointer to the integer 50.

Compilation error if the above portions are considered, since `var` cannot be redeclared as a different variable type.

---

h. ++&addr = &var;

Answer:

The rvalue is `&addr`. The unary `++` (increment) operator requires an lvalue as its operand.

---

i. int* var_addr = &2;

Answer:

Compilation error
The lvalue is `*var_addr` and the rvalue is `2`. The unary `&` (address-of) operator requires an lvalue as its operand.

---

j. var++ = 60;

Answer:

Compilation error
Both `var` and the integer 60 are rvalues in this expression. An lvalue is required as left operand of assignment.
