CIS 343 – Structure of Programming Languages
Homework Assignment #8, Winter 2019
Topic: Subprograms

Due: Friday, April 19, 2019

Student Name: Chandler Scott

1. True or False questions on subprograms:

T / F	Actual parameters are also called arguments.
T / F	In pass by value parameter passing method, changing a formal parameter also changes the corresponding actual parameter (argument).
T / F	Pass by value is an implementation model for *in mode* semantic model of parameter passing.
T / F	Pass by value is the only parameter passing mechanism in C and Java.
T / F	In pass by reference parameter passing mechanism, the formal parameters are really aliases to the corresponding arguments and any changes made to the formal parameter occur to the argument as well.
T / F	Since Java only supports pass by value parameter passing mechanism, we cannot pass references to objects as actual parameters.
T / F	Pass by result parameter passing mechanism is sometimes knows as copy-in, copy-out or copy-restore.
T / F	In pass by reference parameter passing mechanism, an access path to the actual parameter is passed to the called procedure.
T / F	Pass by name parameter passing mechanism performs lazy evaluation of arguments.
T / F	If an actual parameter is a scalar variable, pass by name is equivalent to pass by reference.
T / F	The environment pointer (ep) points to the current activation record.
T / F	The dynamic link is used to implement static scoping.
T / F	Memory allocation for activation records can be performed at load time for languages where functions cannot be nested and recursion is not allowed.
T / F	In a language that uses dynamic scoping, the static/access link is followed to resolve nonlocal references.
T / F	Static link is a link to the defining environment of a function.
T / F	To apply lexical scoping, one must follow the dynamic link to find a nonlocal reference.

2. Determine the output of the following program using the **pass by name** parameter passing method:

```
void main() {
   int j, a[10];
   for (int i = 0; i < 10; i++)
      a[i] = i; // initialize a

   printf("%d", compute(j, a[j], 10, 2));
}


int compute(int x, int y, int n, int step) {
   int sum = 0;
   for (x = 0; x < n, x += step)
      sum = sum + y;
   return sum;
}
```

Output: 14

3. Determine the output of the following program under each of the five parameter passing methods:

```
int i, A[3];

void Q(int B) {
   A[1] = 3;
   i = 2;
   B = 5;
}

void main() {
   i = 1;
   A[1] = 2;
   A[2] = 4;
   Q(A[i]);
   printf("%d %d  %d", i, A[1], A[2]);
}
```

Pass by value: 1 3 4

Pass by result: 2 3 4

Pass by value-result: 1 3 4

Pass by reference: 1 2 4

Pass by name: 1 5 4

4. Show the stack of activation record instances during the second activation of `printx()`. Include both **static** and **dynamic** links in each activation record instance.

```
int x = 4;

void printx(void) {
   printf("%d\n", x);
}

void foo(int y) {
   int x = 4;
   x = x + x * y;
   printx();
}

void main() {
   int z = 3;
   printx();
   foo(z);
}
```

5. Show the stack with all activation record instances, including **static** and **dynamic** links, when the execution reaches **Position 1** in the following skeletal program with **nested procedures**. Assume the procedure Bigsub is at level 1.

   The calling sequence for execution to reach Position 1 in D is as follows: Bigsub calls A, A calls B, B calls A, A calls C, and C calls D.

```
procedure Bigsub is

   procedure A(flag : boolean) is
      ...
      Procedure B() is
      begin
         ...
         A(false);
      end;
      ...
   begin
      if flag then
         B();
      else
         C();
   end;

   procedure C() is
      ...
      Procedure D() is
      begin
         ...      <<---- Position 1
      end;
      ...
   begin
      ...
      D()
      ...
   end;

begin
   ...
   A(true);
   ...
end;
```