Assignment 4

Chapter 9

Due: 4/16/2019

Part A.

1. What are the three general characteristics of subprograms?
2. During execution of subprogram, the calling program should be suspended. It makes there should be only one subprogram execution at a time.
3. Each subprogram has a single-entry point.
4. When the subprogram execution ends, the result and control should always return to the caller.
5. What are the differences between a function and a procedure?

The function should return a value, but a procedure is the way for programs to executes commands. The procedure can be executed by order, because it is a set of command. The function should calculate value based on the input it takes.

1. Describe the ways that aliases can occur with pass-by-reference parameters.

Pass by reference makes access paths available to the called subprograms, thereby providing access to non-local variables. For example, the collisions can occur between actual parameters. Such as **void fun(int &first, int &second)** and **fun(total, total).** It can be said that the name &first and &second are aliases of total. Also, the collisions between array elements also cause aliases. For example in **fun(list[i], list[j]),** if the value of i and j are equal in the list array, then the &first and &second are become aliases as well. The third condition is if two of the formal parameters of subprogram are an element of an array and the whole array, and both are passed by reference, then a call such as **fun1(list[i], list)** could result in aliasing in fun1, because fun1 can access all elements of list through the second parameter and access a single element through its first parameter.

1. Define shallow and deep binding for referencing environments of subprograms that have been passed as parameters.

Shallow binding uses the environment of the call statement that calls the passed subprogram, and most naturally for dynamic-scoped languages.

Deep binding uses the environment of the definition of the passed subprogram, and most naturally for static-scoped language.

1. What is parametric polymorphism?

[Parametric Polymorphism](http://en.wikipedia.org/wiki/Parametric_Polymorphism) is a way to define types or functions that are generic over other types. The genericity can be expressed by using type variables for the parameter type, and by a mechanism to explicitly or implicitly replace the type variables with concrete types when necessary.

1. What is a closure?

A closure is a persistent scope which holds on to local variables even after the code execution has moved out of that block. Languages which support closure (such as JavaScript, Swift, and Ruby) will allow you to keep a reference to a scope (including its parent scopes), even after the block in which those variables were declared has finished executing, provided you keep a reference to that block or function somewhere.

1. Consider the following program written in C syntax:

void swap(int a, int b){

int temp;

temp = a;

a = b;

b = temp;

}

void main(){

int value = 3; list[5] = {10,8,6,4,2};

swap(value, list[0]);

swap(list[0], list[1]);

swap(value, list[value]);

}

For each of the following parameter-passing methods, what are all of the values of the variables value and list after each of the three calls to swap?

1. Passed by value

Value list

First pass 3 [10,8,6,4,2]

Second pass 3 [10,8,6,4,2]

Third pass 3 [10,8,6,4,2]

1. Passed by reference

Value list

First pass 10 [3,8,6,4,2]

Second pass 10 [8,3,6,4,2]

Third pass 4 [8,10,6,10,2]

1. Passed by value-result

Value list

First pass 3 [10,8,6,4,2]

Second pass 3 [8,10,6,4,2]

Third pass 4 [8,10,6,4,3]

1. Consider the following code segment.   
   The function big calls sub1 which calls sub2 which uses x.   
      
    
   1. What would the value of x be with Static scoping?

The value will be 3.

* 1. What would be the value of X be with Dynamic scoping?

The value will be 7.

1. Explain the difference between first, second, and third class values. How does passing a function as an argument fits into these values?

First-class value can be the value passed as an argument, or the value returned from a subroutine, or the value assigned into a variable.

Second-class value just can be passed as an argument.

Third-class value can't be passed as an argument, means the value that can not be pish into the runtime stack.

1. Using C++, Java, or C# write a generic function/method that takes in 2 arguments of any numeric type, adds those 2 arguments together and returns the result.

Java

public<T extends Number> double add (T first, T second)

{

return first.doubleValue()+second.doubleValue();

}