Compiler Principles Assignment: Semantic Analysis & Symbol Tables

Instructions: Based on Chapter 5 of the "Modern Compiler Implementation" presentation, please answer the following questions. All questions are multiple-choice or fill-in-the-blank with a single correct answer.

Question 1 (Multiple Choice)

During the semantic analysis phase, the Symbol Table is crucial. When the compiler exits a scope (e.g., after a function body or a `let` block), it must restore the symbol table to its state before entering that scope. Which of the following implementation methods accomplishes this "restore" operation where the primary overhead involves manipulating a few pointers, rather than performing a large-scale copy of the entire data structure?

A) Making a complete, deep copy of the hash-table-based symbol table upon entering the scope.

B) Using a functional-style balanced binary search tree, which only creates new nodes along the path from the root to the insertion point when adding a new binding.

C) Using an imperative-style hash table combined with a separate "undo stack" to log all modifications, which are then reversed upon exiting the scope.

D) Both B and C are correct.

Question 2 (Fill in the Blank)

In the Tiger pseudocode below, assume the initial symbol table is `σ0 = {g ↦ string, a ↦ int}`. What is the type bound to the variable `a` when the `print(a)` statement on line 8 is executed?

```
1 function f(a:int, b:int, c:int) =
2 (
3   print_int(a+c);
4   let
5     var j := a+b
6     var a := "hello"
7   in
8     print(a);
9     print_int(j)
10  end;
11  print_int(b)
12 )
```

Answer: ____

Question 3 (Multiple Choice)

When implementing an imperative-style symbol table using a hash table with external chaining to resolve collisions, what is the most efficient and scope-compliant way to handle the insertion of a new binding `a ↦ τ2` when an older binding `a ↦ τ1` for the same identifier `a` already exists within the table?

A) Traverse the collision chain to find and overwrite the old binding a ↦ τ1.

B) Append the new binding a ↦ τ2 to the end of the collision chain.

C) Insert the new binding a ↦ τ2 at the head of the collision chain, effectively shadowing the old binding.

D) Throw a "variable redefined" compilation error.

Question 4 (Fill in the Blank)

To improve the efficiency of symbol table operations, particularly the cost of string lookups and comparisons, the Tiger compiler employs an optimization. It converts string identifiers into a unique, fixed-size data type. All subsequent hash calculations and equality checks are performed on this new type, avoiding repeated, character-by-character operations on the original strings. This optimized data type is referred to in the presentation as a ____.

Answer: ____

Question 5 (Multiple Choice)

When processing a language like Java that allows forward references, a member of one class can reference a member of another class that is defined later in the source code. How is this typically handled by the symbol table system to ensure all symbols are visible when needed?

```
// package M;
class E { static int a = 5; }
class N { static int b = 10; static int a = E.a + b; }
class D { static int d = E.a + N.a; }
```

A) A single, global symbol table is used, and all members from all classes are placed directly into it.

B) Each class (E, N, D) has its own symbol table, which are then combined (e.g., σ2 + σ4 + σ6) into a single, comprehensive environment σ7 representing the entire package. This combined environment is then used for compiling the classes within it.

C) The compiler processes declarations in strict sequential order, so when compiling D, it can only see symbols from E and N if they have already been fully analyzed.

D) A stack-based symbol table is used, pushing a class's table onto the stack when entering it and popping it off when exiting.