1. Syntax (10 + 5 + 5 = 20 points)
   The following grammar with start symbol \(\text{\langle hash\rangle}\) defines a language of expressions built from single-letter identifiers and binary operators #, $, and ^:
   \[
   \begin{align*}
   \langle\text{hash}\rangle & ::= \langle\text{hash}\rangle \# \langle\text{dollar}\rangle | \langle\text{dollar}\rangle \\
   \langle\text{dollar}\rangle & ::= \langle\text{dollar}\rangle \$ \langle\text{caret}\rangle | \langle\text{caret}\rangle \\
   \langle\text{caret}\rangle & ::= \langle\text{id}\rangle \^ \langle\text{caret}\rangle | \langle\text{id}\rangle \\
   \langle\text{id}\rangle & ::= 'a' | 'b' | \ldots | 'z'
   \end{align*}
   \]

1a. (10 points) Draw the parse tree for expression \(d\$a#c^a^b\).

1b. (5 points) What is the relative precedence of the three operators?

1c. (5 points) What is the associativity of each of the three operators?

2. Scheme (10 + 10 = 20 points)
   2a. (10 points) Write a recursive Scheme function \texttt{get-last} that returns the last element of a list. For example, \texttt{(get-last '(2 1 5))} should return 5. If the list is empty, \texttt{get-last} should return 0.

   2b. (10 points) Consider the following Scheme function:
      \[
      \text{(define compose (lambda (f g) (lambda (x) (g (f x)))))}
      \]
   Show how Scheme evaluates the expression \((\text{compose} - \text{number->string} 123)\) by rewriting it.
   You should end up with 5 rewriting steps. Note that the built-in function - negates its argument, e.g., \((- 456) \Rightarrow -456\), and the built-in function \texttt{number->string} converts its argument to a string, e.g., \((\text{number->string} 456) \Rightarrow "456"\).

3. Types (10 + 10 = 20 points)
   3a. (10 points) What is the difference between type equivalence and type compatibility?

   3b. (10 points) Explain the distinction between declarations and definitions. Give an example.
4. Scoping (6 + 14 = 20 points)
Consider the following Python program:

```python
def f(x):
    def g(x):
        return h()
    def h():
        return x
    return g(x + 1)
print f(1)
```

4a. (6 points) What does this program print?

4b. (14 points) Show the frames on the stack when \( h \) has just been called. For each frame, show the static and dynamic links. You can either show the links as arrows, or you can use numeric addresses instead if you prefer.

5. Pointers (20 points)
Consider the following C program:

```c
#include <stdlib.h>
int main(int argc, char** argv) {
    double* x = (double*)malloc(5 * sizeof(double));
    int y[5][5];
    double* a = &(x[4]);
    int b = a - x;
    int c = (int*)a - (int*)x;
    int d = (short*)a - (short*)x;
    int (*e)[5] = y + 2;
    int* f = 3 + y[4];
    return 0;
}
```

Assume that this program runs on an architecture where pointers are 8 bytes large, and the sizes of short, int, and double are 4, 8, and 8 bytes, respectively. Further assume that the value of pointer variable \( x \) is the address 500, and the value of pointer variable \( y \) is the address 5,000. What are the values of variables \( a \), \( b \), \( c \), \( d \), \( e \), and \( f \) just before the return statement?

---


Total points: 100.