Name: _____

1. (1 pt.)
   - **Read all material carefully.**
   - This test is **closed book, closed notes**.
   - However, you may refer to **two** standard Letter-sized sheets of paper (both sides) that has **notes hand-written by you**. If used, these sheet of notes must **include your name** near the top and must be **submitted** along with this test.
   - Computing or communication devices of any kind (laptop computers, tablets, phones, calculators, etc.) are not permitted.
   - Network access of any kind (cell, voice, text, data, etc.) is not permitted.
   - Write, and draw, carefully. Ambiguous or cryptic answers receive zero credit.
   - Use class and textbook conventions for notation, algorithmic options, etc.

Print your name clearly in the space provided above.
Do not write anything else on this page.

WAIT UNTIL INSTRUCTED TO CONTINUE TO REMAINING QUESTIONS.

(Do not view any other pages.)

Do not write on this page.
(It is for use in grading only.)

Q	Full	Score
1	1	
2	9	
3	5	
4	10	
5	5	
6	5	
7	5	
8	10	
total	50	

2. (9 pts.) What is the output (or what is the error) when the following Python program is run? **State the exact output clearly. Explain** the answer briefly.

```
def main():
    x = 100

    def f(x):
        def g(y):
            return x

        def h(y):
            print(g(x + 1))
            return x + y

        return h

    print(f(x + 10)(5))


main()
```

3. (5 pts.) Depict the sequence of states of the call stack (runtime stack) when the Python program of Question 2 is run. Explain the answer briefly. Clearly indicate the top of the stack. In each stack frame, clearly write the name of the corresponding function; other details, such as operand stack, are not required.

4. (10 pts.) Using JCoCo assembly language, provide the complete definition of the function `h` from the program of Question 2. (Only the full definition of `h` is needed; other functions do not need to be defined. Explain your answer briefly to qualify for better partial credit.)

5. (5 pts.) **Prove or disprove** this claimed equivalence between regular expressions in Python syntax:

$$\texttt{((a|b)(c|d))*} \equiv \texttt{(ac|ad|bc|bd)*}$$

6. (5 pts.) For the context-free grammar below, clearly state whether the *sentence*

`a b a c c a c a b`

is *valid* (belongs to the language of the grammar). If it is valid then provide a *leftmost derivation* for it *using classroom conventions* (in particular, underlining replaced symbols and annotating arrows with rule numbers); else explain (as precisely as possible) why it is not valid. Ignore all white space. [Hint: It may be easier to answer Question 7 first.]

$$\begin{aligned} S &\rightarrow \texttt{a}\ A \mid \texttt{b} \mid \texttt{c}\ S \\ A &\rightarrow \texttt{b}\ S \mid \texttt{a}\ A \end{aligned}$$

7. (5 pts.) If the sentence of of Question 6 is not valid then make as small a change as possible to yield a valid sentence (else use the unchanged sentence here). Provide a *parse tree* for the (original or modified) sentence.

8. (10 pts.) For each of the following *Standard ML* expressions, provide the response when that expression is evaluated by the `sml` REPL (read-eval-print loop). Assume that the expressions are evaluated in the order listed. In your response, *draw a* ***box*** *around the* ***type*** *and* ***`oval`*** *around the* ***value****.* (If there is an error then clearly explain the error.)

   (a) (2 pts.) `17.29 * 2.0;`

   (b) (2 pts.) `["Hello"] @ ["World"];`

   (c) (2 pts.) `["Hello"] :: ["World"];`

   (d) (2 pts.) `let val f101 = fn x => x + 101;`

   (e) (2 pts.) `let val f201 = fn (x, y, z) => x :: y :: z`

[additional space for earlier material]