Comp 311: B Examination

Problem 1. (15 points) Al Gaulle decides to clean up a program written in a lexically scoped language (such as Scheme, ML, or Pascal) by moving all helper procedures defined at the top level

```
(define helper
  (lambda (...) ...))
```

inside (within the bodies of) the procedures that they help. No helper function is used in more than one procedure.

For example, say that procedures `sort` and `sortHelp` are defined at the top level and that `sortHelp` is only called by `sort`.

```
(define sort
  (lambda (...) ... (sortHelp ...) ... ))

(define sortHelp
  (lambda (...) ...))
```

Al Gaulle's transformation will move the definition of `sortHelp` inside the definition of `sort`.

```
(define sort
  (lambda (...)
    (local
      [(define sortHelp
         (lambda (...) ...))]
      ... (sortHelp ...) ... )))
```

Will this transformation always work (preserve the meaning of the original program)? Why or why not? Justify your answer with a convincing argument or counter-example.

Problem 2. (25 points)

Al Gaulle has designed the ultimate Algol dialect supporting passing parameters by value (using the traditional Algol copying semantics), by name, by reference, and by value-result. Consider the following program written in quasi-algol notation:

```
begin
  int i,j,a[5];     // a is an 5 element array with indices 0-4
  int procedure add(int x, int y)
    begin
      int sum;
```

```
      sum := 0;
      for x := 0 to 4  /* x is the parameter passed to add */
        begin
          a[x] := a[x] + 1;
          sum := sum + y;
        end;
      return sum;
    end;

 for j:= 0 to 4
   a[j] := j;

 i := 0;

 write(add(i,a[i]),a[0]);
end
```

Assume that "write" is a generic print procedure akin to `write` in Pascal and `cout <<` in C++.

What numbers does the program print if both parameters in `add` are passed by:

1. value
2. reference
3. name
4. value-result

Justify your answers by tracing the major steps in each computation.
Note: in Algol, argument evaluation proceeds in left-to-right order.

Problem 3. (10 points) Al Gaulle also wants the new Algol dialect to be type-checked to reduce the number of run-time checks required to ensure safety. Will the choice of parameter passing method affect the type-checking rules? Why or why not?

Problem 4. (25 points) You are given the following Scheme program.

```
(local
  [(define last
     (lambda (l)
       (if (null? (cdr l))
           (car l)
           (last (cdr l)))))
    (define l '((1 2) (3 4 5)))]
  (last (last l)))
```

1. Assume we perform type inference using the type discipline from the simply typed λ-calculus where type the language includes the ground type `int` and the type constructors $\alpha \rightarrow \beta$ and α `list`. Assume that the primitive operations on list have the usual "poly" typings, *e.g.* `car`: α `list` $\rightarrow \alpha$. Is the preceding program typable? Justify your answer.

2. Do we get different results if we use the Hindley-Milner polymorphic generalization of the simple type system? Justify your answer.

Problem 5. (25 points)

1. Convert the program in the preceding problem to CPS. You may assume that `car`, `cdr`, and `null?` are primitive operations that do not have to be converted to CPS form.

2. Change the representation of continuations in your answer to concrete (non-procedure) form.