CMSC 330: Homework Assignment 3

Answer all five questions. Include your OCaml code and explanations. Show your work for the Hoare triple and lambda-calculus questions.

1. Tail recursion, higher-order functions, and tuples

Write a tail-recursive function `select_first pred pairs`. For each pair `(x, y)`, apply `pred` to `y`. If the result is `true`, include `x` in the output list; otherwise, leave it out. Preserve the original order and return `[]` for an empty input list.

Complete the function below. Use the helper `loop` to traverse the list. Every recursive call must be in tail position. You may use `List.rev`, but do not use `List.map`, `List.filter`, or a library folding function.

```
let select_first pred pairs =
  let rec loop acc remaining =
    match remaining with
    | [] -> (* complete this case *)
    | (x, y) :: rest -> (* complete this case *)
  in
  loop [] pairs
```

For example:

```
select_first (fun n -> n >= 10)
  [("A", 12); ("B", 7); ("C", 15)]
(* returns ["A"; "C"] *)
```

The example uses strings and integers, but your function should work with any two component types for which `pred` is valid. The two components of a pair do not have to have the same type.

2. Explain the types in Question 1

Let `x` have type `'a` and `y` have type `'b`. Give the most general type of every item below. Briefly explain how each type follows from your code; do not restrict your answers to the example's strings and integers.

(a) `x`, `y`, `pred`, `pred y`.

(b) `pairs`, `remaining`, `rest`, `acc`.

(c) The complete function types of `loop` and `select_first`.

3. Find the weakest liberal precondition (WLP)

Find and simplify the missing precondition. Start with the postcondition and work backward, showing the substitutions and how you handle both branches. All variables are mathematical integers; ignore arithmetic overflow.

```
{ ? }
if x > 3 then
    y := x - 2
else
    y := x + 4
end if;
y := 2 * y
{ y >= 10 }
```

4. Lambda calculus

Reduce the following expression step by step. Show every beta-reduction, then evaluate the arithmetic. Function application associates to the left: `M N P` means `(M N) P`.

$$(\lambda f.\ \lambda x.\ \lambda y.\ f\ y\ x)\ (\lambda a.\ \lambda b.\ a - b)\ 3\ 10$$

5. OCaml variant types

Consider the following type:

```
type shape =
  | Point
  | Square of int
  | Rectangle of int * int
```

(a) Identify the constructors and state what data each one carries.

(b) What is the type of `Rectangle (3, 5)`? What is the type of the pair `(3, 5)`?

(c) Write `perimeter : shape -> int` using pattern matching. Handle all three constructors using these rules:

- `Point` has perimeter 0.
- `Square s` has perimeter `4 * s`.
- `Rectangle (w, h)` has perimeter `2 * (w + h)`.

Assume all side lengths are positive integers.