CPSC 3740—Fall 2023
Assignment 1

Due: September 26, 2023 11:55pm

Instructions

All written responses must be submitted on Crowdmark. The programming exercises must be submitted on Moodle.

Written Problems

1. (6 marks) Compute the weakest preconditions of the following statements for the given postconditions. Assume all variables are integers.

   (a) (2 marks)

   ```
   a = 2*b + 1;
   b = a*a + 1;
   { a > 10 }
   ```

   (b) (4 marks)

   ```
   if (b > 0)
     a = 4*b - 10;
   else
     a = b + 172;
   { a > 0 }
   ```

2. (8 marks) Consider the following grammar:

   $$S \rightarrow a \mid b \mid S + S \mid S - S$$

   (a) (2 marks) Describe in English the strings that are in the language defined by the grammar.

   (b) (4 marks) Show a parse tree for `a + b - a` (spaces added for clarity).

   (c) (2 marks) Is this grammar ambiguous? Explain.

3. (12 marks) Consider the following C++ code segment:

   ```
   int d = 20;

   int f(int b)
   {
     static int c = 0;
     c *= b;
     return c;
   }

   int main()
   {
     int a;
     cin >> a;
     cout << f(a) * d << endl;
   }
   ```

   For each variable `a`, `b`, `c`, `d`:

   - identify all type bindings and storage bindings
   - for each binding, determine when the binding occurs
   - identify the scope and lifetime

Programming Exercises

In the next two assignments, you will learn parts of the Racket programming language on your own. The reference for the language can be found at `https://docs.racket-lang.org/reference/index.html`. Section numbers below refer to this reference site.

You can find `drracket` on the Linux machines on the department's computers. Write your solutions to the two exercises in `as1.rkt` and submit that file on Moodle. Your work will be graded both on correctness and documentation. Please document each function as well as the overall algorithm.

One of the basic data types is a list (Section 4.10):

- Empty lists are denoted `()`. It is also called a null list (`null?`)
- A non-empty list consists of a first element (`car`) and the rest (`cdr`) (Section 4.9)
- Lists can be recursively nested.
- Lists can contain "atoms" that are not lists. e.g. integers such as 1, 2, 3.

For example, the following are lists:

- `(1 2 3)`
- `()`
- `(1 (2 3) (4 5 (6)))`

Note that in Racket, lists are considered as function calls (first argument is the function name and the rest are parameters). To avoid this behaviour we use the single quote. For example, `'(1 2 3)`.

To define a function in Racket, you use `define` (Section 3.14). For example, the following is a function that counts the number of elements in a given list `x`:

```
(define (listcount x)
  (if (null? x)
    0
    (+ 1 (listcount (cdr x)))))
```

Note: this function does not work if `x` is not a list.

1. (10 marks) Write a function called `mymember?` that will take an object `x` and a list `y`, and return `#t` (true) if `x` is an element of `y` and `#f` (false) otherwise.

   Note that `y` may be a nested list, but `x` is only a member of `y` if `x` is one of the top-level elements.

   For example:

   ```
   (mymember? '(4 5 (6)) '(1 2 3 (4 5 (6))))
   ```

   should return true, while

   ```
   (mymember? '4 '(1 2 3 (4 5 (6))))
   ```

   should return false.

   Use the built-in `equal?` to determine if two elements are equal.

2. (10 marks) Write a function called `myappend` that will take a list `x` and an object `y`, and return a new list consisting of the elements of `x` followed by `y`.

   For example:

   ```
   (myappend '(1 2 3) '4)
   ```

   should return `(1 2 3 4)`, while

```
(myappend? '(1 2 3) '(4 5 6))
```

should return `(1 2 3 (4 5 6))`.

3. (10 marks) Write a function called `myunion` that will take two lists `x` and `y`, and return a list that has the elements of `x`, followed by the elements of `y` that are **not** found in `x`.

   For example:

   ```
   (myunion '(1 3 5 7 9) '(1 2 3 4 5 6 7))
   ```

   should return `(1 3 5 7 9 2 4 6)`.

   Hint: use the functions defined in the previous parts.