

# CPTS 355 Sample Midterm 2 Fall 2009

NAME \_\_\_\_\_

**Directions:** Answer all of the questions. The test is open-book, open notes. You may **not** use a computer, PDA, calculator or phone during the test. Remove all caps or face their visors to the back.

**Write your name on your notes sheet and turn it in along with your exam.**

Make sure that you have a complete exam **now**.

Write your name on your exam **now**.

1 **Directions:** Short answer – each answer is an expression or short phrase.

a) What is the ML type of  $[(3, \text{"hi"})]$ ?

b) What is the ML type of  $([3], \text{"hi"})$ ?

**Directions:** The following are short answer questions. Write your answer in the space provided.

2) What is the *type* of the ML function created by the declaration

```
fun f ([], v) = []  
  | f (x::xs, v) = (x, v) :: (f(xs, v));
```

3) Write C declarations corresponding to the following two ML declarations. The first two are straightforward. The last will take some problem-solving thinking.

a) `type person = {lastName: string, firstName: string, birthDate: int}` (hint: a struct)

b) `datatype Tree = Elm | Birch | Fir;` (hint: an enum)

c) `datatype Tree =  
 Num of int |  
 Node of {left: Tree, right: Tree, oper: (int*int)->int}`  
(hint: a union)

4) Determine the weakest precondition for the following program fragments, using the assignment rule. Show your work by showing the derivation of the weakest precondition step by step according to the sequence and assignment rules – visible, organized step-by-step work is **required** for full credit.

a) `z = 4; n = z * 5 + n; {z == n - 30}`

b) `z = 3; y = y + (z - 5); {y == 6*z + 1}`

5) In the following program, assume that  $n \geq 1$ .

```

i = 1;
j = n;
s = 0;
while (i<j) do
    s = s + i + j;
    i = i + 1;
    j = j - 1;
end

```

( $\wedge$  stands for logical “and”)

Prove that  $\{(i \leq j+1) \wedge (i+j=n+1) \wedge (s = (i-1)*(n+1))\}$  is an invariant of this loop by showing that

$$\{(i \leq j+1) \wedge (i+j=n+1) \wedge (s = (i-1)*(n+1) \wedge (i<j))\} \Rightarrow$$

```

wp(s=s+i+j; i=i+1; j=j-1;,
    {(i <= j+1) ^ (i+j=n+1) ^ (s = (i-1)*(n+1))}
)

```

Solution: calculate the wp then check to see whether the implication holds:

```

wp(s=s+i+j; i=i+1; j=j-1;,
    {(i <= j+1) ^ (i+j=n+1) ^ (s = (i-1)*(n+1))}
)
==
wp(s=s+i+j; i=i+1, {(i<=j-1+1)^(i+j-1=n+1)^(s=(i-1)*(n+1))})
==
wp(s=s+i+j, {(i+1<=j)^(i+1+j-1=n+1) ^ (s=(i+1-1)*(n+1))})
==
{(i<j) ^ (i+j=n+1) ^ (s+i+j)=(i*(n+1))}
==
{(i<j) ^ (i+j=n+1) ^ (s+ (n+1))=(i*(n+1))}
==
{(i<j) ^ (i+j=n+1) ^ (s=(i-1)*(n+1))}

```

Okay, so now we know the wp of the invariant and the loop body. Does the invariant together with the loop test imply it? That is does

$$\{(i \leq j+1) \wedge (i+j=n+1) \wedge (s = (i-1)*(n+1) \wedge (i<j))\} \Rightarrow$$

$$\{(i<j) \wedge (i+j=n+1) \wedge (s=(i-1)*(n+1))\}$$

YES! Because each of the conjuncts in the rhs of the implication appears in the lhs.

6a) Define a recursive ML function, `pairprods`, that will take a list of two-element sublists as input and produce as output a list whose elements are the products of the elements of each of the sublists. For example, (`pairprods [[1,4], [14,5], [7,3], [22,6]]`) should produce `[4, 70, 21, 132]` as its answer.

6b) `Pairprods` can also be defined using the built-in higher-order function `map`. To define `pairprods` using `map` you would write a definition like

```
fun pairprods2 L = map pairfn L
```

Write a definition for `pairfn` so that `pairprods2` computes the same thing that `pairprods` computes.

7) Draw a derivation tree for the sentence

$$3 * 4 - 5 - 2$$

using the grammar given below. Assume that `<e>` is the start symbol.

```
<e> → <t> * <e> | <t>
<t> → <f> - <t> | <f>
<f> → 3 | 4 | 5 | 2
```

What are the **terminal symbols** of this grammar? \_\_\_\_\_

Which operator, + or \*, has higher precedence? \_\_\_\_\_

What is the value of this expression when computed according to your parse tree? \_\_\_\_\_

In this grammar \* is \_\_\_\_\_ associative.

9) Consider again the datatype

```
datatype Tree =
```

```
  Num of int
```

```
  | Node of {left: Tree, right: Tree, oper: (int*int)->int}
```

of problem 3c. Tree can be used to represent arithmetic expressions.

For example, `Node{left=(Num 1), right=(Num 2), oper=(op +)}` represents the expression `1+2`. **Note:** `(op +)` here just means the function implemented by the `+` operator which is usually used in infix notation. `(op +)` is a way to use the addition function when infix is not appropriate.

Write a recursive ML function, `eval`, having type `Tree->int` that evaluates a value of type `Tree` to determine the value of the arithmetic expression that it represents.