HW3

10/30 out, 11/13 due

1 Consider the abstract syntax of a small language

Com ::= Exp; | while Exp do Com | Com; Com

Exp ::= Num | Var | Exp Op Exp | Var = Exp

Op ::= + | - | * | / | %

Num ::= Digit | Num Digit

Digit ::= 0 |1 | … | 9

Var :: = *left unspecified*

This is similar to the language given in the lecture on denotational semantics, except that "assignment" is now treated as expression rather than command and a new "expression command" Exp; is added.

Give a denotational semantics for this language. For simplicity, you need only give the semantic functions for the blue-colored rules. (The semantic functions for all the other rules remain unchanged.) (20%)

Notes

1 The semantic function $\mathcal{C}$ remains the same type $\mathcal{C}$: Com $\rightarrow$ Store $\rightarrow$ Store$_\perp$
But, the semantic function $\mathcal{E}$ now has type
$\mathcal{E}$: Exp $\rightarrow$ Store $\rightarrow$ (Store $\times\mathbb{Z}$)$_\perp$ where (Store $\times\mathbb{Z}$)$_\perp$ = Store $\times\mathbb{Z} \cup \{\perp\}$
That is, the meaning of an expression in a store is either undefined ($\perp$) or a pair (s,v), where s $\in$ Store and $v \in \mathbb{Z}$

2 The intended meanings of assignment expressions and expression commands are exactly that of C.

3 For Exp Op Exp, evaluate the left Exp first.

4 Define the semantic functions carefully.

2 [Palindrome] (20%)

Write a SML function

palindrome : int -> bool

to check if a nonnegative integer reads the same forward and backward.

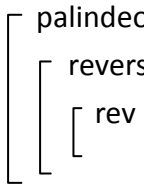
Requirements

1 The function palindrome shall contain a local function reverse to reverse the number, i.e. reverse 123 $\Rightarrow$ 321, and check if n is equal to reverse(n).

2 The function reverse shall contain a local recursive function rev and use it to reverse the number, as follows:
reverse 123 $\Rightarrow$ rev 123 0 $\Rightarrow$ rev 12 3 $\Rightarrow$ rev 1 32 $\Rightarrow$ rev 0 321 $\Rightarrow$ 321

2 (Continued)

In summary, the three functions shall be nested as follows:



Sampe run

– palindrome 12321;

val it = true : bool

– palindrome 12345;

val it = false : bool

3 [Mergesort] (20%)

Write a SML function

msort : int list -> int list

to sort a list of integers by mergesort.

Note: It is inefficient to split a list in the middle. (Why?) Think out a better way of splitting a list into two sublists.

Sample run

– msort [5,8,1,7,3,4,2,6];

val it = [1,2,3,4,5,6,7,8] : int list

4 [Higher-order function, Church numeral] (20%)

A natural number n may be represented by the higher-order function $\lambda f.\lambda x.f^n\ x$, called the nth Church numeral. For examples, 0, 1, 2, 3 are represented by the Church numerals $\lambda f.\lambda x.x$, $\lambda f.\lambda x.f\ x$, $\lambda f.\lambda x.f\ (f\ x)$, $\lambda f.\lambda x.f\ (f\ (f\ x))$, respectively.

You are asked to write the following four ML functions, where $\bar{n}$ denotes the nth Church numeral.

a) n2c converts a natural number to the corresponding Church numeral.
   n2c $n \Rightarrow \bar{n}$

b) c2n converts a Church numeral to the corresponding natural number.
   c2n $\bar{n} \Rightarrow n$

c) ++ is an infix operator for adding two Church numerals.
   $\bar{m}$ ++ $\bar{n} \Rightarrow \overline{m+n}$

d) ** is an infix operator for multiplying two Church numerals.
   $\bar{m}$ ** $\bar{n} \Rightarrow \overline{mn}$

4 (Continued)

Requirements

1 Both ++ and ** are left associative. ** is of precedence level 7, and ++ precedence level 6.

2 Define $\overline{m}$ ++ $\overline{n}$ and $\overline{m}$ ** $\overline{n}$ directly. Do NOT convert $\overline{m}$ and $\overline{n}$ to m and n, respectively, and then convert $m + n$ back to $\overline{m+n}$

Sample run

– (c2n o n2c) 7;
val it = 7 : int
– c2n (n2c 3**n2c 4++n2c 5);
val it = 17 : int

5 [Concrete data type] (20%)

Given a string that represents a postfix arithmetic expression that is formed by single-digit numbers and operators +,-,*,/,and %, write a SML function

eval : string -> int

that uses a stack to compute the value of the postfix expression. You may assume the string represent a legal postfix expression.

Requirements

1 First, use the function explode to turn a string into a list of characters.
– explode "23+";
val it = [#"2",#"3",#"+"] : char list
Next, process the list of characters (that represents a postfix expression).

2 Define a polymorphic concrete stack in ML as
datatype 'a stack = Empty | Push of 'a*'a stack;
For examples,
– Push(2,Push(1,Empty));
val it = Push (2,Push (1,Empty)) : int stack
– Push(#"2",Push(#"1",Empty));
val it = Push (#"2",Push (#"1",Empty)) : char stack

Sample run

– eval "23+";
val it = 5 : int
– eval "234+73/73%-+*";
val it = 16 : int