CSE 305: Pattern Matching and Data Types

Due: March 3, at 11:59 pm

1 Overview

In this assignment you are asked to implement some basic functions with a specified behavior and type (or signature) working with a data type representing *printable* data which is defined as follow:

```
type printable = B of bool   | U |
                 S of string | L of int list | P of printable * printable
```

This definition is also provided in the file *printer.ml* which you can use as a basis for your development.

A printable element can be a bool, a string, a list of integers, a U (more on this shortly), or a pair of printable elements (notice that the definition of `printable` is recursive). Here some examples of `printable` values:

```
ex0 = B true
ex1 = S "Schrute bucks"
ex2 = U
ex3 = L [1; 8; 5; 0; 3]
ex4 = P (P (U, P (P (L [1; -5; 13], U), P (L [0], B true))),
        S "Hello")
ex5 = P (P (B false, P (P (L [-21; 53; 12], S "c"), P (L [0], B false))), S "Hello")
ex6 = P (P (U, P (P (L [15; -15; 213], S "c"), P (P (U, U), S "false"))),
       S "Hello")
ex7 = P (S "Stanley nickels", L [1000000])
```

We will use some of these examples to illustrate the functions that you will implement.

Also note that constructor names ***must begin with a capital letter*, and binding names ***must begin with a lowercase letter*** (or an underscore).

2 Function specifications

The first function you need to implement is `count_u` whose behavior is to recursively traverse a printer value, `p`, and count how many printable elements inside `p` are of the form `U`. Recall that `U` is an empty constructor whose set of values consists of one element, namely `U` itself. We can give to this function the following type:
`count_u: (p: printable) -> int`.
For example, if we run `count_u` on ex2, ex3, ex4, ex5, ex6 we have:

```
count_u ex2 = 1
count_u ex3 = 0
count_u ex4 = 2
count_u ex5 = 0
count_u ex6 = 3
```

The second function you need to implement is `global_or` whose behavior is to compute the logical or of all the booleans inside a printable value `p`. There are two "corner cases" that your solution needs to consider: 1) if there is exactly one Boolean in the printable `p` then `global_or` should return this value as result, 2) if there is no Boolean in `p` then, since we cannot compute an or, `global_or` should return a value `None` asserting that no information is available. The type we want for `tostring` is then the following:
`global_or: (p: printable) -> bool option`.
For example, if we run `global_or` on ex0, ex4, ex5, ex6 we have:

```
global_or ex0 = Some true
global_or ex4 = Some true
global_or ex5 = Some false
global_or ex6 = None
```

The third function you need to implement is `f_on_int_list` which should traverse a printable `p` and apply a given function `f` to all the elements of each list in `p`. The type we want for `f_on_int_list` is the following:
`f_on_int_list: (f: int->int) -> (p: printable) -> printable`.
For example, if we run `f_on_int_list` on ex3, ex4, ex5, ex6, ex7 with some auxiliary anonymous functions we have:

```
f_on_int_list (fun t-> t*t + 1) ex3 = L [2; 65; 26; 1; 10]
f_on_int_list (fun _-> 42) ex4 = P (P (U, P (P (L [42; 42; 42], U), P (L [42], B true))),
S "Hello")
f_on_int_list (fun t-> 2*t) ex5 = P (P (B false, P (P (L [-42; 106; 24], S "c"),
    P (L [0], B false))), S "Hello")
f_on_int_list (fun t -> t mod 2) ex6 = P (P (U, P (P (L [1; -1; 1], S "c"),
    P (P (U, U), S "false"))), S "Hello")
f_on_int_list (fun t -> t +1) ex7 = P (S "Stanley nickels", L [1000001])
```

You may find helpful here to use the function `List.map`.

The fourth function you need to implement is `sum_all_ints` which returns the sum of all the integers in an input printable `p`. There is a particular case that your solution needs to consider: if there is no integer `sum_all_ints` should return a value `None` asserting that no information is available. The type we want for `sum_all_ints` is the following:
`sum_all_ints: (p: printable) -> int option`.
For example, if we run `sum_all_ints` on the following examples we have:

```
sum_all_ints (B true) = None
sum_all_ints (L [1; 2; 3]) = Some 6
sum_all_ints (f_on_int_list (fun t-> 2*t) ex5) = Some 88
sum_all_ints (f_on_int_list (fun t-> t*t) ex6) = Some 45819
```

The last function you need to implement is `tostring` which converts a printable element `p` into a string, following a pre-order traversal of the tree. That means when you are converting printable element of the form `P(p1,p2)` to a string, you will need to concatenate the recursive calls as `(tostring p1)^(tostring p2)`. The type we want for `tostring` is the following:
`tostring: (p: printable) -> string`. **When converting values of U to strings, use "U".** For example, if we run `tostring` on our examples we have:

```
tostring ex0 = "true"
tostring ex1 = "Schrute bucks"
tostring ex2 = "U"
tostring ex3 = "18503"
tostring ex4 = "U1-513U0trueHello"
tostring ex5 = "false-215312c0falseHello"
tostring ex6 = "U15-15213cUUfalseHello"
tostring ex7 = "Stanley nickels1000000"
```

You may find convenient in your solution to first define a function `string_of_intlist` converting a list of integers into a string which can then be used in the definition of `tostring`.

3 Submission Instructions

Late submissions will not be accepted. You can use Autolab to confirm your program adheres to the specification outlined. Only your last Autolab submission will be graded for you final grade. This means you can submit as many times as you want. If you have any questions please ask well before the due date.

3.1 Autolab account creation

An Autolab account has been already created for you. You can access Autolab at: `https://autograder.cse.buffalo.edu`. If you already have an account for Autolab from a previous course or another course you are taking this semester you can log in normally and you should see CSE305 in your list of courses. If you have not used Autolab before you will need to go to: `https://autograder.cse.buffalo.edu/auth/users/sign_in` and click on the Forgot your password? link. The following page will prompt you for your email address. Use your UB email address with your UBIT id as this is the email used to create your account for you. You will receive an email with instructions on how to reset your password and log into Autolab. It is important you verify your account is working well in advance of the turn in date.

3.2 Autolab submission instructions

When you log into Autolab and pick the CSE305 course you will see an assignment called Printer. The total of points for the assignment is: 60. On the Autolab page of Printer, click Submit and choose your source file. Your program should be an .ml file. You can submit as many times as you wish before the deadline.

3.3 Additional Resources

Additional Resources For information on how to run the various programming language compilers/interpreters:

- `https://wiki.cse.buffalo.edu/services/content/ocaml`
- `https://caml.inria.fr/pub/docs/manual-ocaml/stdlib.html`
- `https://caml.inria.fr/pub/docs/manual-ocaml/`

You will find resources for these languages on the Piazza course web site, under the Re- sources page. You might also find the following page, listing available CSE systems, helpful too: `https://wiki.cse.buffalo.edu/services/content/student-systems`.