CSE 305: Pattern Matching and Data Types

Due: 11:59 pm Tuesday, February 23, 2021

1 Overview

In this assignment you are asked to implement portions of the checkout system backend for the Oakland, California Municipal Library (OCaML). You will be writing functions to handle calculating fines for overdue items. Your implementations must adhere to a specified behavior and type (or signature). These functions will operate on a set of three nested data types representing the library's records.

The data types representing records are defined as follows:

```
type catalog_item = Book of string * string * string
                  | Movie of string * int * string
                  | CD of string * string * string
                  | Computer
type checkout_entry = Item of catalog_item
                  | New of checkout_entry | Extend of checkout_entry
                  | Pair of checkout_entry * checkout_entry
type cart = CartEntry of checkout_entry * int option * cart | Empty
```

These definitions are also provided in the file *library.ml* which you can use as a basis for your development.

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

Here a `catalog_item` represents one physical item in the library. For a book, the parameters are the title, author, and publisher. For a movie, the parameters are title, year, and director. For a CD, the parameters are the album name, artist, and length. Computer represents using a public computer, and thus has no arguments.

Some items are new or can only be checked out as a pair with another item. A `checkout_entry` represents one checked out bundle. Finally, a `cart` combines a `checkout_entry`, an integer option, and another cart. The integer representing the number of days overdue. If the option is `None`, the item is not overdue. If the option is `Some n`, n is the number of days the item is overdue. You may assume n is always positive. If there is nothing in the cart, then it should be Empty.

2 Function specifications

2.1 `string_of_item : catalog_item -> string`

The first function you will implement converts a `catalog_item` to a `string`. Books shall be rendered as "*TITLE* by *AUTHOR* (*PUBLISHER*)". A movie shall be "*TITLE* (*YEAR*) by *DIRECTOR*". A

CD shall be rendered as "*ALBUM* by *ARTIST*". Finally, computers should be "Public Computer". You may find the OCaML function `string_of_int : int -> string` helpful.

Example Outputs

```
# string_of_item (Book ("Types and Programming Languages", "Benjamin Pierce",
  "The MIT Press"));;
- : string = "Types and Programming Languages by Benjamin Pierce (The MIT Press)"
# string_of_item (Movie ("The Imitation Game", 2014, "Morten Tyldum"));;
- : string = "The Imitation Game (2014) by Morten Tyldum"
# string_of_item Computer;;
- : string = "Public Computer"
```

2.2 `string_of_entry : checkout_entry -> string`

The second function you must implement converts `checkout_entry` to a `string`. An normal `Item` shall be rendered just as the wrapped `catalog_item` would be. A new entry should be rendered as "(NEW) *ENTRY*", where *ENTRY* is the wrapped `checkout_entry`. A extend entry should be rendered as "(EXT) *ENTRY*", where *ENTRY* is the wrapped A pair should be "*ENTRY1* and *ENTRY2*", where *ENTRY1* and *ENTRY2* are the items. In implementing this function, you may wish to call the `string_of_item` function you previously wrote.

Example Outputs

```
# string_of_entry (Item(Book("Types and Programming Languages", "Benjamin Pierce",
  "The MIT Press")));;
- : string = "Types and Programming Languages by Benjamin Pierce (The MIT Press)"
# string_of_entry (New (Item (Movie ("The Imitation Game", 2014,
  "Morten Tyldum"))));;
- : string = "(NEW) The Imitation Game (2014) by Morten Tyldum"
```

2.3 `daily_fine : checkout_entry -> float`

The third function you must implement computes the fines (in dollars) incurred for each day an item is overdue. The fines are as follows:

- **Books**: 0.25
- **Movie**: 0.50
- **CD**: 0.50
- **Computer**: 0.00
- **New Items**: Double the normal fine
- **Extend Items**: Triple the normal fine
- **Pair**: The sum of the fines for each item

2.4 `total_fine : cart -> float`

The fourth function you will implement takes a `cart` and returns the total fine for all items. Remember, if the `int option` is `None`, the item is not overdue and thus no fines are due for that item.

You may find `float_of_int : int -> float` helpful in implementing this function.

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 Library. The total of points for the assignment is 80. On the Autolab page of Library, 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.

4 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`.