COMP 410
Spring 2018

Data Structures in Mercury

1.) Consider the following grammar, representing arithmetic expressions:

$$i \in Integer$$

$$e \in Expression ::= i \mid \oplus e \mid e_1 \otimes e_2$$

$$\oplus \in UnaryOperation ::= \textbf{negate}$$

$$\otimes \in BinaryOperation ::= \textbf{plus} \mid \textbf{times}$$

Write Mercury type definitions below, corresponding to the above grammar. Try to make your type definitions match the grammar as closely as possible. Multiple answers are possible.

2.) Consider the definition of inductive lists, shown below:

$$e \in ListElement$$

$$\ell \in List ::= \texttt{cons}(e, \ell) \mid \texttt{nil}$$

Write Mercury type definitions below, corresponding to the above list definition. List elements should be treated as a generic type.

3.) Write a Mercury procedure below which will extract all the numbers from a list which are greater than five. An example query is shown below:

```
?- extract([9, 2, 6, 5], List).
List = [9, 6].
```

Be sure to write appropriate `pred` and mode annotations. You may assume the first parameter will always be given. As a hint, you will need to use `if, then, else`.

4.) The `filter` procedure generalizes the pattern of extracting elements from a list. This works by parameterizing the **computation** used to determine if an element should be in the output list. An example query is shown below, which ends up performing the same operation as `extract` above:

```
?- filter([9, 2, 6, 5],
          (pred(Num::in) is semidet :- Num > 5),
          List).
List = [9, 6].
```

To assist you, `pred` and `mode` annotations for `filter` have already been provided. Implement the `filter` procedure below, in Mercury. As a hint, you will need to use `if, then, else`, along with `call` to call the passed procedure.

```
:- pred filter(list(T), pred(T), list(T)).
:- mode filter(in, in(pred(in) is semidet), out) is det.
```