Assignment II (35 pts)

Burak Ekici

Assigned : April the 3rd, 23h55
Due : April the 16th, 23h55

1 Prelims

It is possible scan integer lists in an ocamllex (`lexer.mll`) implementation as folows:

```
type token =
      | BLANK : token
      | LPAREN: token
      | RPAREN: token
      | COMMA : token
      | EOL   : token
      | NUM   : int → token
```

```
rule tokenize = parse
  | ' '
    { tokenize lexbuf }
  | '['
    { LPAREN }
  | ']'
    { RPAREN }
  | ','
    { COMMA }
  | eof
    { EOL }
  | [ '0'-'9' ]+ as i
    { NUM (int_of_string i) }
  | _ as c
    { raise (BadChar c) }
```

The rules listed as part of the `tokenize` function map strings represented by the regular expressions into the collection of tokens inhabited by the `token` type. For instance, the rule "'[' { LPAREN }" tokenizes the symbol '[' into the token LPAREN. Similarly, the rule "' ' { tokenize lexbuf }" skips the input blank symbols simply by moving onto the next symbol delivered by the input buffer `lexbuf`.

One can then collect tokens in an OCaml list out of an input string thanks to `getTokensFromString` function.

```
let rec getTokensFromBuffer(b: lexbuf): token list =
  let tkn = tokenize b in
    match tkn with
      | EOL → [EOL]
      | t   → t :: getTokensFromBuffer b

let getTokensFromString(s: string): token list = getTokensFromBuffer(from_string s)
```

2 Tasks

The task is to develop a parser employing the menhir generator in the way detailed below.

Q1 (15 pts). Construct in the `ast.ml` file a new data type `llist` together with functions taking an (or a pair of) `llist` instance(s) `l` (and `m`), and computes

the pretty printer	`let rec printLlist(l: llist): unit`
the length	`let rec length(l: llist): int`
the append	`let rec append(l: llist) (m:llist): llist`
the reverse	`let rec reverse(l: llist): llist.`

Q2 (20 pts). Implement an `llist` parser, in the `parser.mly` file, employing menhir realizing the context free grammar $G = \big(\{S, T\}, \text{integers} \cup \{[,]\}, P, S\big)$ with

$$P = \left\{ \begin{array}{lcl} S & \rightarrow & [\,] \mid [T] \\ T & \rightarrow & \text{integers}, T \mid \text{integers} \end{array} \right\}.$$

Do not forget to implement following functions in the `main.ml` file:

```
let switchTokens(t: Lexer.token): Parser.token
let astOfString(s: string): llist
```

Expected behavior of the entire implementation when invoked inside the "`let main: unit`":

`let l1 = astOfString "[ 7, 98,4 ,8 , 89 ]" in printLlist l1;`	prints	[7,98,4,8,89]
`let l2 = append l1 l1 in printLlist l2;`	prints	[7,98,4,8,89,7,98,4,8,89]
`let l3 = reverse l2 in printLlist l3;`	prints	[89,8,4,98,7,89,8,4,98,7]
`let v = length l3 in printf "%d" v;`	prints	10.

Nota Bene (in general).

1. receive support from helper functions if needed;
2. files inside the attached `list.zip` archive must be the starting point;
3. implement your functions inside `ast.ml`, `parser.mly` and `mail.ml` files, and submit them back under `list.zip` archive.

Important Notice:

- Collaboration is strictly and positively prohibited; lowers your score to 0 if detected.
- Any submission after 23h55 on April the 16th will NOT be accepted. Please respect the deadline!