MAC0316/5754

Exercício Programa 1

Data de entrega: 11/10/2021

Instruções:

1. Você deve entregar seu programa pelo eDisciplinas em um **único arquivo** *.rkt* contendo as definições do interpretador
2. Programas **atrasados** terão uma **penalidade de 20%** na pontuação **POR DIA**.

Interpretador (10 pontos)

a) **(6 pontos)** Implemente `let`, `let*` e `letrec` como visto em classe. Suporte um número fixo de argumentos (1, 2 e 1, respectivamente). Let, let* devem ser implementados como açúcar sintático (apenas como exprS) e letrec deve ser implementado como função primitiva (exprC).

b) **(2 pontos)** Implemente a primitiva `quote`, que gera um símbolo, com `'` no seu interpretador. Assim:

```
(interpS '(quote alan))
```

deve retornar

```
- Value
(symV 'alan)
```

Para isso você deve incluir um novo tipo de valor.

c) **(2 pontos)** Implemente a primitiva `read-loop` inicia um conjunto de leituras a partir do terminal e interpreta cada uma delas, indicando o resultado. Assim:

```
(interpS '(read-loop))
```

deve avaliar todas as expressões simbólicas entradas pelo usuário no terminal. A leitura deve se encerrar quando usuário digitar a expressão '#END. Note que este arquivo irá conter "expressões normais", não chamadas ao interpretador ou ao parser. Isso significa que você mesmo deve implementar essas chamadas (como ilustrado no exemplo acima). Veja abaixo um pequeno exemplo de uma funçãozinha que lê um conjunto de expressões até a indicação de final

```
(define read-till-end (lambda ()
  (let ( (input (read)))
    (if (and (s-exp-symbol? input ) (eq? (s-exp->symbol input) '@END))
        (display 'FINISHED-INTERPRETER)
        (begin (display "\nintepret-command:")
               (display input)
               (display "\nresult:")
               (display 'um-resultado)
               (display "\n")
               (read-till-end))))))
```

IMPORTANTE: Não será dado **crédito parcial** aos itens acima. Apenas aqueles que funcionarem receberão pontos (e.g. se você implementar `let` no parser mas não no interpretador, a implementação do `let` está incompleta).

IMPORTANTE: Você deve usar a versão `plai-typed` do interpretador com fechamentos *closureTypedNoStore.rkt* (disponibilizado com este enunciado). Assim, você poderá utilizar os comandos de gerenciamento de arquivos do Scheme. (**Dica:** procure a documentação do comando `display`).