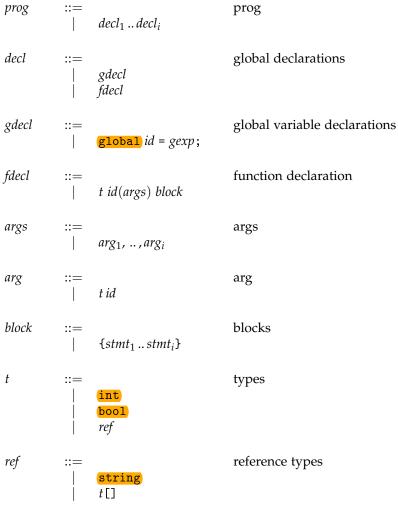
## Oat v. 1 Language Specification

CIS341 - Steve Zdancewic

March 19, 2020

## 1 Grammar

The following grammar defines the Oat syntax. In the grammar, id denotes an identifier, n denotes a non-negative integer, and s denotes a string literal. All binary operations are left associative with precedence levels indicated numerically. Higher precedence operators bind tighter than lower precedence ones.



```
| fty
```

```
global initializers
gexp
                      п
                      tnull
                      true
                      [new t[] \{gexp_1, ..., gexp_i\}]
stmt
               ::=
                                                                     statements
                      lhs = exp;
                      vdecl;
                      return exp;
                      return;
                      id(exp_1, ..., exp_i);
                      if_stmt
                     for(vdecls; exp_opt; stmt_opt) block
                      while(exp) block
if_stmt
                                                                     if statements
                      if(exp) block else_stmt
else_stmt
                                                                     else
                      \epsilon
                      else block
                      \verb"else" \textit{if\_stmt}
lhs
                                                                     lhs expressions
               ::=
                      id
                      exp_1[exp_2]
vdecls
                                                                     decl list
               ::=
                      vdecl_1, ..., vdecl_i
vdecl
                                                                     local declarations
               ::=
                      var id = exp
```

```
ехр
        ::=
                                          expressions
              id
              n
              S
              t null
              true
              false
              exp_1[exp_2]
              id(exp_1, .., exp_i)
              new t[]{exp_1, ..., exp_i}
              newt[exp_1]
              exp_1 bop exp_2
              иор ехр
               (exp)
                                          (left associative) binary operations
bop
        ::=
                                             precedence 100
                                             precedence 90
               +
                                             precedence 90
               <<
                                             precedence 80
                                             precedence 80
              >>
              >>>
                                             precedence 80
                                             precedence 70
               <
               <=
                                             precedence 70
                                             precedence 70
              >
              >=
                                             precedence 70
                                             precedence 60
               ! =
                                             precedence 60
                                             precedence 50
                                             precedence 40
               [&]
                                             precedence 30
                                             precedence 20
               [ ] ]
иор
                                          unary operations
               !
```