0/7 Questions Answered

Quiz 3 from Spring 2021 (Practice)

STUDENT NAME

Search students by name or email...

Q1 Ambiguity

4 Points

Consider the following program:

```
S -> aSa | M
M -> aM | T
T -> Ta | ε
```

Use the derivation of the string "aa" to show why the above grammar is ambiguous:

Enter your answer here

Save Answer

Q2 Parsing Logistics

4 Points

Q2.1

2 Points

Consider the following grammar:

```
Expr -> Expr LogicalOp Expr | Bool
LogicalOp -> and | or
Bool -> true | false
```

This grammar **cannot** be parsed by a recursive descent parser.
Clearly explain why not.

Enter your answer here

Save Answer

Q2.2
2 Points

Consider the following grammar:

```
Expr -> Expr LogicalOp Expr | Bool
LogicalOp -> and | or
Bool -> true | false
```

This grammar **cannot** be parsed by a recursive descent parser.
Clearly explain how we can fix the grammar above so that it **can** be parsed by a recursive descent parser.

Enter your answer here

Save Answer

Q3 Regular and Non-regular CFGs
2 Points

Consider the language represented by the grammar from problem 1, duplicated for your convenience:

```
S -> aSa | M
M -> aM | T
T -> Ta | ε
```

Which of the following regular expressions represents the CFG above?

- ○ a{4}a{4}
- ○ a*
- ○ S+aT?
- ○ This CFG cannot be represented by a regular expression
- ○ None of the above

Save Answer

Q4 Regular CFG Design
3 Points

Write a CFG for the following language (given as a regular expression): `a+b?`

Enter your answer here

Save Answer

Q5 Nonregular CFG Design

3 Points

Write a CFG that accepts strings of the form $a^x b^z c^y$ where $x \geq 0, y \geq 0, z = 2x + y$. (Hint: The most common solution involves starting with something of the form S -> AB)

Enter your answer here

Save Answer

Q6 Parsing

4 Points

Consider the following grammar:

```
S -> aSe | abT
T -> cT | dMk
M -> f | k
```

A parser for this language would rely on the following functions:

```
let match_toks toks x =
match toks with
| y :: t when y = x -> t
| _ -> raise (InvalidInputException "bad match")

let lookahead toks =
match toks with
| h :: t -> h
| _ -> raise (InvalidInputException "empty")
```

The TAs tried creating a parser for it, but got stuck. Help us fill in the blanks!

```
let parse_S toks=
        let toks = match_tok toks “a” in
        match lookahead toks with
        | “b” -> let toks = match_toks toks “b” in
                 | ___ A ___ |
        | “a” -> let toks = | ___ B ___ | in
                 match_toks toks “e”

and let parse_T toks =
        match lookahead toks with
        | “c” -> let toks = match_toks toks “c” in
                 parse_T toks
        | “d” -> | ___ C ___ |

and let parse_M toks =
        match lookahead toks with
        | “f” -> match_toks toks “f”
        | “k” -> match_toks toks “k”
```

A

Enter your answer here

B

Enter your answer here

C

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission