

Family Name:..... Other Names:

Student ID:..... Signature

COMP 261 : Test 1

14 March 2024,

Instructions

- Time allowed: **50 minutes**
- Attempt **all** the questions. There are 50 marks in total.
- Write your answers in this test paper and hand in all sheets.
- If you think a question is unclear, ask for clarification.
- This test contributes 15% of your final grade.
- You may use dictionaries and calculators.
- You may write notes and working on this paper, but make sure your answers are clear.

Questions

Marks

1. Grammars and Parse Trees

[15]

2. Coding a Parser

[25]

3. Ambiguous Grammars

[10]

TOTAL:

Question 1. Grammars and Parse Trees**[15 marks]**

Consider the following grammar that describes a made-up language for advertising rental properties.

In this grammar

- Non-terminals are in uppercase; terminals are enclosed in quotation marks,
- [...] means one or more repetitions of what is in the brackets.
- NUM matches any terminal that is a non-negative integer.

```
AD ::= [ PROPERTY ]+
PROPERTY ::= "(" SIZE "," RENT ")"
SIZE ::= NUM "br"
RENT ::= "$" NUM
NUM ::= matches "[0-9]+"
```

(a) **[5 marks]**

The following three sentences are almost, but not quite valid sentences of the grammar above. For each sentence, circle the first token where a parser could identify the error.

Please note that space is used as the delimiter in Scanner.

(i) (3 br , \$ 800 , 2 br , \$ 400 , 2 br , \$ 500)

(ii) (2 br , \$ 500) , (1 br , \$ 300) , (2 br , \$ 300)

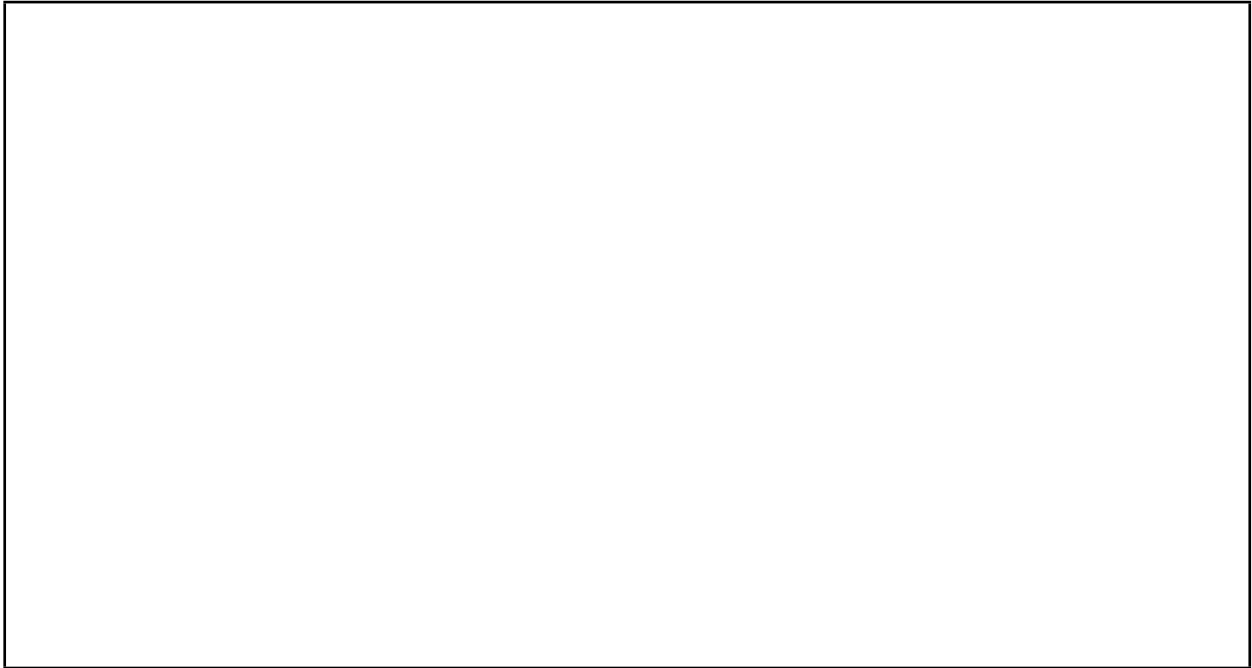
(iii) ((1 br , \$ 200) (2 br , \$ 300) (2 br , \$ 350))

(Question 1 continued on next page)

(Question 1 continued)

(b) [5 marks] Draw the Concrete Parse Tree of the following sentence according to the grammar.

(1 br , \$ 250) (3 br , \$ 600)



(c) [5 marks] Abstract Syntax Trees.

Draw the Abstract Syntax Tree for the sentence in the previous sub-question.



Question 2. Coding a Parser**[25 marks]**

Suppose you are writing a parser for the grammar in question 1.

```
AD ::= [ PROPERTY ]+
PROPERTY ::= "(" SIZE "," RENT ")"
SIZE ::= NUM "br"
RENT ::= "$" NUM
NUM ::= matches "[0-9]+"
```

Your parser program (below) already includes some constants, three parse methods, and utility methods (`require(..)` and `fail(..)`).

The two sub-questions are:

- (a) to write more parse methods so that the top parse method should return an Abstract Syntax Tree of ANodes, or throw an exception if a sentence is invalid. We provide methods for `parseSize(..)`, `parseRent(..)`, `parseNum(..)` and they all return `int`, so you do not need to create nodes for `SIZE`, `RENT` and `NUM`.
- (b) to write the `AdNode`, `PropertyNode` classes defining the different kinds of ANodes.

You are to complete the methods in Java, not pseudocode.

```
//----- constants (patterns) -----
static final Pattern BR_PAT = Pattern.compile("br");
static final Pattern DOLLAR_PAT = Pattern.compile("\\$");
static final Pattern LEFT_PAT = Pattern.compile("\\(");
static final Pattern RIGHT_PAT = Pattern.compile("\\)");
static final Pattern NUM_PAT = Pattern.compile("[0-9]+");
static final Pattern COMMA_PAT = Pattern.compile("\\,");
//----- parse... methods -----
/** Parses the rule:  SIZE ::= NUM "br"           please note it returns an int */
public int parseSize(Scanner s){
    int num = parseNum(s);
    require(BR_PAT, s);
    return num;
}
/** Parses the rule:  RENT ::= "$" NUM           please note it returns an int */
public int parseRent(Scanner s){
    require(DOLLAR_PAT, s);
    int rent = parseNum(s);
    return rent;
}
public int parseNum(Scanner s){
    if (s.hasNext("[0-9]")) {return s.nextInt();}
    fail("Expecting integer"); return -1;
}
//----- Utility methods -----
public void require(Pattern pat, Scanner s){
    if (s.hasNext(pat)) {s.next(); return;}
    fail("expecting "+ pat);
}
public void fail(String msg){ System.out.println(msg); throw new RuntimeException(msg);}
```

(Question 2 continued on next page)

(Question 2 continued)

(a) [15 marks] Complete the `parseAd(..)`, `parseProperty(..)` methods below.

```
/** Parses the rule:      AD ::= [ PROPERTY ]+ */
public ANode parseAd(Scanner s){

}

/** Parses the rule:  PROPERTY ::= "(" SIZE "," RENT ")"      */
public ANode parseProperty(Scanner s){

}

}
```

(b) [10 marks] Define the ANode classes

The parser in the previous sub-question needs the AdNode, PropertyNode classes to create the Abstract Syntax Tree of ANodes. Suppose we want to print out the "average rent per room for all properties in one advertisement" for a valid sentence, for example,

```
Parser parser = new Parser();
Scanner s = new Scanner("( 2 br , $ 200 ) ( 3 br , $ 900 )");
System.out.println ( parser.parseAd(s).getAveRentPerRoom());
```

should print out 200, which is calculated as: $(200/2 + 900/3) / 2$

You will need to define a `getAveRentPerRoom()` method for each kind of ANodes.

Grammar:

```
AD ::= [ PROPERTY ]+
PROPERTY ::= "(" SIZE "," RENT ")"
SIZE ::= NUM "br"
RENT ::= "$" NUM
NUM ::= matches "[0-9]+"
```

```
//----- Node classes -----
```

```
interface ANode{  
    public int getAveRentPerRoom();  
}
```

```
class AdNode implements ANode{
```

```
}
```

```
class PropertyNode implements ANode{
```

```
}
```

Question 3. Ambiguous Grammars**[10 marks]**

Ambiguous grammars (where a text can have multiple different parse trees) cannot be parsed by deterministic top-down recursive descent parsers, like the parsers described in the lectures.

The following grammar is ambiguous.

```
BLOCK ::= CELL | BLOCK "stack" BLOCK | BLOCK "concat" BLOCK
CELL ::= matches "c[0-9]+"
```

(a) **[4 marks]** Draw two alternative parse trees of the following sentence according to this grammar:

c10 stack c5 concat c2

Tree #1

Tree #2

(Question 3 continued on next page)

(Question 3 continued)

(b) [6 marks] Rewrite the rule for BLOCK (and any additional rules you need) so that the grammar covers the same language but is no longer ambiguous. To get full marks, you need to make sure that the "concat" operation has higher priority (precedence), for example, c5 and c2 should be concatenated in the previous example.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.