Homework 1

Due September 28, 2023

Q1: Language standards

1. (JLS20, Section 8.1.1.2., p.242) A class can be declared sealed if all its direct subclasses are known when the class is declared, and no other direct subclasses are desired or required. A class can be declared final if its definition is complete and no subclasses are desired or required.
2. (JLS20, Section 8.3, p.266) A class can inherit multiple fields with the same name from superclasses or superinterfaces; by using the interface name (InterfaceName.FieldName) and super keyword (super.FieldName) to distinguish between the inherited fields.
3. No, it can lack an intialiser. (JLS20, Section 4.12.4, p.102) A blank final is a final variable whose declaration lacks an initializer.
4. The idea behind definite assignment is that an assignment to the local variable or blank final field must occur on every possible execution path to the access. (JLS20, Chapter 16, p.713)
5. (ECMA-334, section 13.9.5, p.232) The C# foreach loop is specifically for direct iteration over elements of a collection as opposed to being for more general and custom iteration; also foreach forbids modification to the iteration variable in the embedded statement (such as i++ or i-- in the for loop).
6. (ECMA-334, Section 15.5.5, p.286) An uninitialised variable is given the default value of the variable's type, e.g. uninitialised booleans are by default false.
7. (ECMA-334, Section 9.2.4, p.60) A dynamic variable isn't checked until runtime, and an exception will be thrown if there is an error. The dynamic keyword allows for dynamic binding
8. Other formats other than the .class file are acceptable, but they must be able to be mapped into the class file format by a class loader written in Java. (JLS20, Section 13.1 p. 452)
9. Java's alphabet/character set is Unicode. All code must be written in characters found in Unicode on systems that support only ASCII. (JLS20, Section 1.1, p.2)

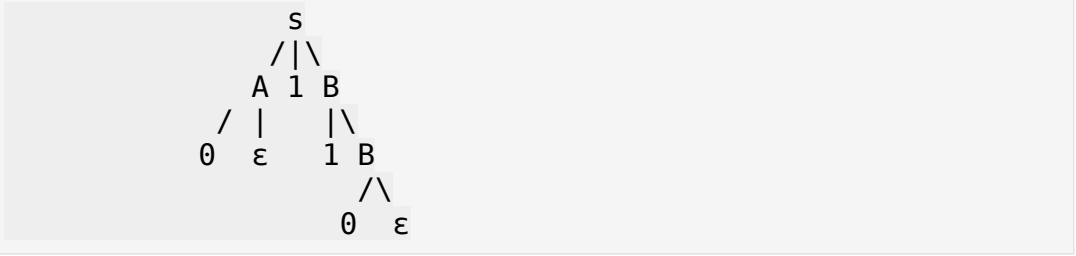
Q2: Grammars and parse trees

1.
    - a. The language whose string contains either 0 or more instances of the following: one or more instsances of a followed by a valid string followed by an optional b, or b followed by a valid string followed by one or more instances of a; or nothing.
    - b. The language whose string contains of one 0 with any equal number of 0s and 1s on either side.
    - c. The language whose strings contains a's followed by b's followed by c's followed by d's, with an equal number of one or more pairs of a's and d's, and an equal number of zero or more pairs of b's and c's.

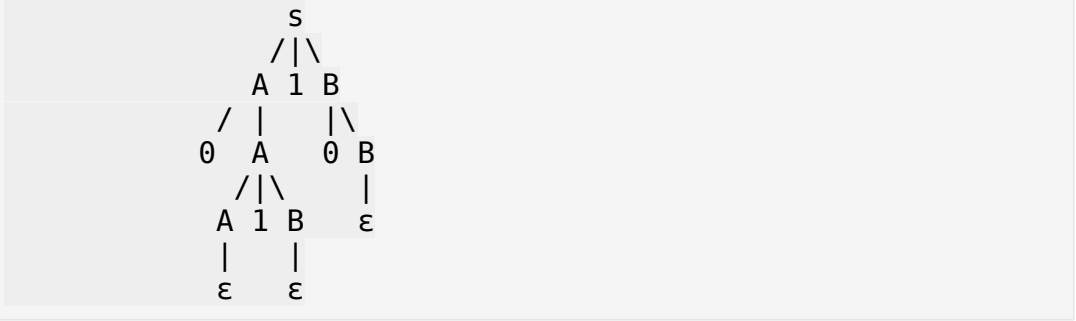
- d. The language whose strings contain n number of a's, followed by zero or more b's, followed by 2n number of a's.
- e. The language whose strings are palindromes containing zero or more a's and b's.

2.

- a. 0* (1[01]*)* 1 [01]*
- b. String: 0110
  - Parse tree 1:



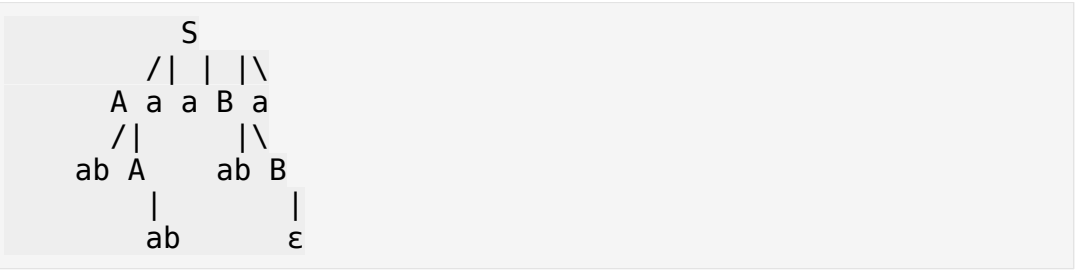
  - Parse tree 2:



- c.

3.

- a. (ab)+aa(ab)*a
  - S -> A aa B a
  - A -> ab A | ab
  - B -> ab B | ε
- b. ab ab aa ab a



4.

- N3 includes all non-terminals of all 3 languages.

- G3's grammar includes all grammar rules of G1 and G2; S3->S1; S3->S2; if all terminal symbols are the same, then L3 is a union of all non-terminals and grammar rules of L1 and L2.

Q3: Regular expressions

1. [ac]*b[ac]*
2. [aaabc]*
3. ([abc]{3})*
4. ([1-9][0-9]*)?[02468]
5. 1*(0*11+0*)*0*1**
6. [a-zA-Z>/]*[a-zA-Z<>]*([a-zA-Z>]/*)*

Q5: Associativity and precedence

1.
   - ODD 32 ADD 3 SWP REV 24
   - ODD 32 ADD 3 SWP (REV 24)
   - (ODD 32) ADD 3 SWP 42
   - 2 ADD (3 SWP 42)
   - (2 ADD 423)
   - 425
2.
   - REV 421 SUB 1 ADD 4 SWP ODD 32
   - (REV 421) SUB 1 ADD 4 SWP ODD 32
   - 124 SUB 1 ADD 4 SWP (ODD 32)
   - 124 SUB 1 ADD (4 SWP 2)
   - 124 SUB (1 ADD 24)
   - (124 SUB 25)
   - 99
3.
   - ODD 725 SWP REV 78 SWP ODD 242 ADD 2
   - ODD 725 SWP (REV 78) SWP ODD 242 ADD 2
   - (ODD 725) SWP 87 SWP ODD 242 ADD 2
   - 2 SWP 87 SWP (ODD 242) ADD 2
   - 2 SWP (87 SWP 242) ADD 2
   - (2 SWP 24287) ADD 2
   - (242872 ADD 2)
   - 242874
4.
   - 71 CAT 6 SWP 12 SUB ODD REV 423
   - 71 CAT 6 SWP 12 SUB ODD (REV 423)
   - 71 CAT 6 SWP 12 SUB (ODD 324)
   - 71 CAT (6 SWP 12) SUB 24
   - (71 CAT 126) SUB 24

- (71126 SUB 24)
- 71102

5.
- ODD (10 SWP ODD REV 921 SWP 12)
- ODD (10 SWP ODD (REV 921) SWP 12)
- ODD (10 SWP (ODD 129) SWP 12)
- ODD (10 SWP (2 SWP 12))
- ODD (10 SWP 122)
- ODD (12210)
- 220

Q6.

1. "Hello Darling! Hello World!"
   - g() && (i() || f()) && (h() || i() && f()) = false
   - g(False) && (i(True) || f(True)) && (h(Either) || i(True) && f(True)) = false
   - don't want to print h() so i() && f() must be true
   - the second i() won't be printed
   - the whole statement msut be false so "what lovely weather" won't print, so with the previous two statements evaluating to true g() must be false
   - f(): true;
   - g(): false;
   - h(): either;
   - i(): true;
2. (n4944 C++, Section 7.6, p.150) In C++, the logical operators such as && and || guarantees left-to-right exaluation, and the right hand side isn't evaluated if the left is already true.
3. In the case where for example a variable might be null or some value that would cause an error/exception, it would be useful to hide it later in the evaluation sequence so that it can be guaranteed never to evaluate on an error-raising value.
   - E.g. if(arr != null && arr.length > 3). In this case, if the array is null, it wouldn't try to find its length as it would raise an error.

Q7: Bindings and nested subprograms

Unit	Var	Where declared
main	a b	main main
sub1	a b	sub1 main
sub2	a b c	sub2 main sub2
sub3	a b c d	sub2 sub3 sub2 sub3