Question 1

Which of the following statements is correct?

- ○ In our abstract grammar, VarDecl is a subclass of VarDeclStmt.
- ◉ In our abstract grammar, Block is a subclass of Stmt.
- ○ In our abstract grammar, Stmt is a subclass of Block.
- ○ In our abstract grammar, LHS is a subclass of RHS.

Question 2

Which of the following statements is correct?

- ○ In our abstract grammar, ArithComprExpr is a subclass of CompExpr and BinaryExpr is a subclass of ArithCompExpr.
- ◉ In our abstract grammar, ArithCompExpr is a subclass of CompExpr and CompExpr is a subclass of BinaryExpr.
- ○ In our abstract grammar, BinaryExpr is a subclass of CompExpr and ComprExpr is a subclass of ArithCompExpr.
- ○ In our abstract grammar, BinaryExpr is a subclass of CompExpr and ArithComprExpr is a subclass of CompExpr.

Question 3

Which of the following statements is correct?

- ○ 1. In our abstract grammar, FunctionName is a subclass of Call.
- ○ 2. In our abstract grammar, Expr is a subclass of Stmt.
- ○ 3. In our abstract grammar, Expr is a subclass of ReturnStmt.
- ○ 4. In our abstract grammar, Operand is a subclass of Expr.
- ◉ 5. None of the above is correct.

Question 4

Which statements are correct?

- [x] In our abstract grammar, Name is a field of class TypeDeclaration.
- [ ] In our abstract grammar, IntTypeName is a field of class Typename.
- [ ] In our abstract grammar, Expr is a field of class ExprStmt.
- [x] In our abstract grammar, Expr is the type of a field of class ExprStmt.

Question 5

Which of the following statements is correct?

1. Attributes/methods for name analysis (name lookup) compute the types of the variable/parameter/function names.
2. Attributes/methods for name analysis (name lookup) search the declaration nodes of the names of variables/functions/types in the enclosing block.
3. Attributes/methods for name analysis (name lookup) search the declaration nodes of the names of variables/functions/types in the AST.
4. Attributes/methods for name analysis (name lookup) check for scoping errors.

Question 6

Consider the following statements

a. Lookup for types (lookupType) works exactly the same as field lookup.
b. Lookup for variables (lookupVar) works exactly the same as function lookup.
c. Lookup for functions (lookupFunction) works exactly the same as field lookup.
d. Lookup for types (lookupType) works exactly the same as function lookup.

Which ones above are correct?

1. Statements a), b) and c).
2. Statements b), c) and d).
3. Statements c), d) and a).
4. Statements d), a) and b).

Question 7

Consider the following statements

a. VarName.namecheck() checks if a variable name is undeclared.
b. VarDecl.namecheck() checks if a variable name has multiple declarations.
c. VarName.namecheck() checks if a variable name has multiple declarations.
d. VarDecl.namecheck() checks if a variable name is undeclared.

Which ones above are correct?

- (●) 1. Statements a) and b).
- ( ) 2. Statements c) and d).
- ( ) 3. Statements a), b), c) and d).
- ( ) 4. None of the statements.

Question 8

Which of the following statements is correct?

- ( ) 1. A node of type TypeDescriptor is attached to AST nodes by the parser.
- ( ) 2. A node of type TypeDescriptor is attached to AST nodes by the semantic analyser.
- ( ) 3. A node of type IntType/BooleanType/VoidType/ArrayType is attached to AST nodes by the parser.
- (●) 4. A node of type IntType/BooleanType/VoidType/ArrayType is attached to AST nodes by the semantic analyser.

Question 9

Consider the following statements

a. A type descriptor node `IntType` will be attached to each of the AST node AddExpr, SubExpr, MulExpr, DivExpr and ModExpr in an AST.
b. The same descriptor node `IntType` will be attached to each of the AST node AddExpr, SubExpr, MulExpr, DivExpr and ModExpr in an AST.
c. The type descriptor node `IntType` will be attached to each of the AST node `IntTypeName` in an AST.
d. The type descriptor node `IntType` will be attached to each of the AST node `VarDecl` in an AST.

Which ones above are correct?

- ( ) 1. Statements a) and c).
- (●) 2. Statements b) and c).
- ( ) 3. Statements a) and d).
- ( ) 4. Statements b) and d).

Question 10

Which of the following statements is/are incorrect?

- [ ] 1. AddExpr, SubExpr, MulExpr, DivExpr and ModExpr are defined to be of INT type in our language.
- [ ] 2. LtExpr, GtExpr, LeqExpr and GeqExpr are defined to be of BOOLEAN type in our language.
- [ ] 3. EqExpr and NeqExpr are defined to be of BOOLEAN type in our language.
- [x] 4. ArrayLiteral is defined to be of INT type in our language.
- [ ] 5. IntLiteral is defined to be of INT type in our language.

Question 11

Consider the following statements

a. The type check of a source program starts with the leaf nodes of an AST and after the children nodes are checked, the parent node will be checked, until the root node.
b. The type check of a source program starts with the method Program.typecheck().
c. The type check of a source program starts with the root of an AST and the typecheck() method of an AST node calls the typecheck() method of its children node.
d. The typecheck() method of leaf nodes in an AST does not do anything.

Which ones above are correct?

(

- ○ Statements a), b) and c).
- ◉ Statements b), c) and d).
- ○ Statements c), d) and a).
- ○ Statements d), a) and b).
- ○ Statements a), b), c) and d).

Question 12

In the following set of tests for return statements, which case is not tested?

```
@Test
public void test_a() {
      runtest(false,
                  "module M {" +
                  "  int foo() {" +
                  "    return true;" +
                  "  }" +
                  "}");
}
@Test
public void test_b() {
      runtest("module M {" +
                  "  void foo() {" +
                  "    return;" +
                  "  }" +
                  "}");
}
@Test
public void test_c() {
      runtest("module M {" +
                  "  int foo() {" +
                  "    return 0;" +
                  "  }" +
                  "}");
}
@Test
public void test_d() {
      runtest(false,
                  "module M {" +
                  "  void foo() {" +
                  "    return 0;" +
                  "  }" +
                  "}");
}
```

- ○ the function is of void type and returns nothing;
- ○ the function is of a non-void type and returns an expression of the correct type;
- ○ the function is of void type but returns an expression of some other type;
- ◉ the function is of a non-void type but returns nothing;
- ○ the function is of a non-void type but returns an expression of a type different from the function type.

Question 13

Choose the most appropriate answer : In order to generate the complete class definitions for all AST nodes, the input to the tool Jastadd are

- ○ Context free grammar and abstract grammar
- ◉ Abstract grammar, attribute grammar and inter-type declarations and methods
- ○ Attribute grammar and inter-type declarations and methods
- ○ Context free grammar, abstract grammar and attribute grammar
- ○ Abstract grammar and inter-type declarations and methods

Question 14

In the following source programs, which one has semantic error when a name is declared more than once? (you should choose at least one).

- [ ]
```
module M {
 void foo() {
          int x;
         { boolean x;  }
     }
}
```
- [x]
```
module M {
   void foo(int x;   boolean x) { }
}
```
- [ ]
```
module M {
     int x;
    void foo() {
         boolean x;
     }
}
```
- [ ]
```
module M {
      import N;
      int x;
}
module N {
      public int x;
}
```

5, 8,9,10