MONTANA STATE UNIVERSITY

PROGRAMMING LANGUAGES

Midterm

SPRING, 2018

Name: ______________________________

Student Id: ______________________________

Question	Points	Score
1	30	
2	2	
3	2	
4	2	
5	2	
6	2	
7	10	
8	15	
9	15	
10	20	
Total:	100	

1. (30 points) Mark each question as either True (T) or False (F).

   (a) ◯ T ◯ F C++ is considered by many to be the first object-oriented programming language.

   (b) ◯ T ◯ F COBOL was designed as a programming language for business applications.

   (c) ◯ T ◯ F LISP was originally developed for Artificial Intelligence problems.

   (d) ◯ T ◯ F A strongly typed programming language is one that detects all type errors at either compile time or run time.

   (e) ◯ T ◯ F A non-terminal that occurs on the far left of any production rule is left associative.

   (f) ◯ T ◯ F The "Von Neuman" architecture is still used for the basis of most computers today.

   (g) ◯ T ◯ F Any finite language can be defined by a regular expression.

   (h) ◯ T ◯ F A grammar with a finite set of terminal symbols can only define a finite language.

   (i) ◯ T ◯ F EBNF was designed to make context free grammars easier to read. It does not offer any more descriptive power than BNF.

   (j) ◯ T ◯ F Associativity rules only apply to operators of the same precedence.

   (k) ◯ T ◯ F The order of production rules is not significant, i.e., two grammars with identical rules but given in a different order will always define the same language.

   (l) ◯ T ◯ F Overloading functions and operators can only be done in dynamically typed languages because of their greater flexibility.

   (m) ◯ T ◯ F An advantage of using an interpreted language is that it runs faster than compiled code.

   (n) ◯ T ◯ F A programming language that uses dynamic scoping allows programs to see every symbol table on the stack.

   (o) ◯ T ◯ F The procedural programming paradigm treats procedures as first class objects.

2. (2 points) When two variable identifiers are allowed to share the same address pointer it is known as:

   A. dynamic binding
   B. dynamic scoping
   C. aliasing
   D. weak typing
   E. dynamic typing

3. (2 points) Context free grammars are used to capture which aspect of a programming language?

   A. lexical structure
   B. syntax
   C. type system
   D. operational semantics
   E. axiomatic semantics

4. (2 points) Choose the (one) example of orthoganality from the statements below

   A. An array element can be any data type except void or a function.
   B. All statements (including assignment, if, and while) return some value.
   C. Parameters are passed by value, unless they are arrays, in which case they are pased by reference.
   D. Structures (but not arrays) may be returned from a function.
   E. Function passing is not permitted.

5. (2 points) In Java, a method declared in an interface is implicitly:

   A. static
   B. private
   C. final
   D. abstract
   E. none of the above

6. (2 points) Circle all of the following that are Object Oriented languages.

   A. C#
   B. Fortran 77
   C. Java
   D. LISP
   E. Perl

7. Consider the following code in some C-style language. Assume the `writeln` function prints to standard output followed by an `EOL`.

```
// global variables
int foo = 3;
int bar = 5;

void printmult(int bar) {
    writeln foo * bar;
}

void addbar(int bar) {
    foo = foo + bar;
    printmult(foo);
}

void test() {
    int foo = 2;
    int bar = 7;
    printmult(4);

    foo = 1;
    bar = 13;
    addbar(9);
    writeln foo;
}
```

(a) (5 points) What does the function `test` print if the language uses static scoping?

(b) (5 points) What does the function `test` print if the language uses dynamic scoping?

8. Answer these questions about the following grammar. The starting symbol is `<re>`.

```
<do> ::= <me>
<do> ::= <do> % <me>
<re> ::= <do> & <re>
<re> ::= <do>
<me> ::= ( <re> )
<me> ::= a | b
```

(a) (2 points) Which operator has higher precedence?

(b) (2 points) What is the associativity of the `%` operator?

(c) (2 points) What is the associativity of the `&` operator?

(d) (2 points) Does this grammar describe a finite or an infinite language?

(e) (2 points) Is this grammar ambiguous?

(f) (5 points) Draw a parse tree to explain the string:

`b & a % (a & b) & a`.

9. (15 points) complete the following Java loop skeleton below. Your code should match street addresses of a certain format and print out the information in the manner described below.

   Using regular expressions, match street addresses of the format: a number followed by one or more words and ending in the string "st", "dr", or "ave". For example, your regex should be able to return "1212 pine st". You do not need to worry about all of the other possible road types, just the three listed above.

   The following is an example input:

```
Thomas Aquinas, (555) 555-5555, 1212 Pine St Bozeman, MT 59715
Bram Stoker, (123) 456-7890, 123 W College Dr, Helena, MT 59601
Ben Kingsley, (321) 123-6756, 9876 N 13th Ave, Nome, AL 99762
```

```
// This loop processes some text file line by line
// You only need to complete the code within the loop

while(my $line = <STDIN>) {

    // Coverts all text to lower case
    $line = lc($line);









} // End while loop
```

10. (20 points) Consider the following Java code:

```
interface Inter {
   int fun();
}
interface Face {
   int work();
}
abstract class Picaso {
   int blue = 42;
   String period() { return "."; }
   abstract int paint();
}
class Reach extends Picaso implements Inter {
   public int paint() { return 3; }
   public int fun() { return 5; }
   int up(int input) { return input * 2; }
}
class Arm extends Reach implements Face {
   Arm(int wave) { blue = wave; }
   public int work() { return 7; }
   public int fun() { return blue + 2; }
   String up(String input) { return input + period(); }
}
```

Now consider this concrete implementation of two classes:

```
public static void main(String[] args) {
   Reach r1 = new Reach();
   Reach r2 = new Arm(10);

   // Expressions executed here
}
```

Give the value for each expression below or "error" if an error would occur.

(a) r1.fun()

(b) r2.fun()

(c) r1.blue

(d) r2.blue

(e) r2.work()

(f) r2.paint()

(g) r1.up(3)

(h) r2.up(12)

(i) r2.up("7")

(j) r1.up("16")

THIS PAGE INTENTIONALLY LEFT BLANK