COSC4315 Midterm Exam
Multiple Choice #2

1. Instructions

The exam is individual. You can use your computer and any Internet resource. You cannot ask TAs or professor any clarification. Each question is worth 10 points. Choose one answer. Write one question # and one chosen letter per line in your answer file.

2. Questions

1. Consider the functional expression: add(multiply(2,4),add(1,3)). which is the correct posfix expression?

   (A) 2 4 multiply 1 add 3 add
   (B) 2 4 multiply 1 3 add add
   (C) 2 4 multiply 1 add 3 add
   ~~(D) 2 multiply 4 add add 1 3~~
   ~~(E) 2 4 1 3 add multiply add~~

   https://www.mathblog.dk/tools/infix-postfix-converter/

2. Which features in Python generally produce variable mutation?

   (A) function definition, initialization, while loop
   (B) function definition, recursive function, while loop
   (C) variable assignment, a sort function, while loop
   (D) variable assignment, pass parameter by value, for loop
   (E) function definition, pass parameter by reference, initialization

3. Consider for(), while() and do/while() loops in C++ and Java. Which one is false?

   ✓(A) they have the same syntax in both languages
   (B) do/while() is more general than while()
   ✓(C) a while() loop my not terminate
   ✓(D) a for() is less bug-prone
   ✓(E) iterators are object-oriented while() loops

4. Consider char * strings in C (< $string.h$ >) and new strings in C++ (< $string$ >).
   I. the string capacity must be stored in C++ False
   II. they are compatible data types
   III. A null terminator is required in C

   (A) I
   (B) I,II
   (C) I,III
   (D) II,III
   (E) I,II,III

5. Consider the recursive function definition below. Assume head() and tail() are defined. Which statement is false? in Python.

```
def search(l,e):
  if len(l)==0:
    return []
  elif head(l)==e:
    return [head(l)]
    else:
      return search(tail(l),e)
```

✓ (A) it always returns a list
✓ (B) it works correctly with sorted or unsorted lists
(C) it does not produce mutation
(D) it ignores duplicates
(E) it gives an incorrect answer with empty lists

6. Which one is false about a binding? a programming language?

In computing, a binding is an application programming interface (API) that provides glue code specifically made to allow a programming language to use a foreign library or operating system service (one that is not native to that language).

✓ (A) It connects a variable name and a data type
✓ (B) It associates a variable name and a memory cell location
✓ (C) It is automatically manipulated by an interpreter or compiler
✓ (D) It establishes a link between a calling function and a called function ex: main & sort()
(E) It is unnecessary in a functional language, but required in other languages
Python

7. Compare C, C++ and Python. Which one is true?

✗ (A) Python does not support functional programming
✗ (B) C++ does not support functional programming
(C) Python data types are inspired by C data types
(D) Python syntax is more complex than C++ syntax
✗ (E) C and C++ discourage library-based programming

8. Which if the following grammars for arithmetic expressions is not ambiguous?

~~(A) $E \rightarrow E + E, E \rightarrow E * E$~~
~~(B) $E \rightarrow E + E|T, T \rightarrow E * T|T$~~
(C) $E \rightarrow T + T|T, T \rightarrow E * T|T$
(D) $E \rightarrow T + E|T, T \rightarrow F * T|F$
(E) $E \rightarrow E + F|F, E \rightarrow E * E$

ambiguous: grammar produces sentential form & has two or more parse trees.

https://www.cs.oberlin.edu/~bob/cs331/Class%20Notes/February/February%208/AmbiguityAndPrecedence.pdf

https://imgur.com/a/3p4TGxh
Screenshots from PPT.

9. Which of following λ calculus expressions is invalid?

(A) $(\lambda \underline{xy}.yx)\lambda x.xz$
✓ (B) $(\lambda y.xy)\lambda y.y$
(C) $(\lambda x.x)1$
✓ (D) $(\lambda x.\lambda y.xy)\lambda y.y$
(E) $(\lambda x.yx)0$ x'0

https://projectultimatum.org/cgi-bin/lambda

10. Which is the regular expression for a reserved word in C++?

```
(A)  letter+letter*
(B)  letter(letter|digit)+
(C)  letter letter+
(D)  letter*letter+
(E)  letter*
```

Identifiers ⇒ letter(letter + digit)+
IntegerNumeral ⇒ digit digit *
Real Numeral ⇒ digit + .digit + E (epsilon |+|-) digit +

"|" ⇒ union
{"ab", "c"} + {"d", "e"}
⇒ {"ab", "c", "d", "e"}

\d = any digit (0-9)

. = any char

* = 0 or more
(is a quantifier)

.*= "sample".* "wild card"
(+ any char)

* Zero or more Ocurrences —
ab*c ⇒ 'ab', 'abc', 'abbc'
1 ∞ 1

+ one or more occurance
ab+c ⇒ 'abc', 'abbc', 'abbbc'
1 ∞ 1