**ICS-365 Exam #1**

**75 Points**

1. **Make sure you read the questions carefully and answer each part.**
2. Some questions may be answered with as little as 2 sentences. Some may require more. No question should require more than 6 sentences, but use whatever space you feel necessary.
3. Upon completion of exam, make sure to upload it to D2L.
4. You will be graded on completeness and quality of your answers. So it is possible to have an answer with the correct information, but not be complete.
5. Use the Word Highlighter function to answer the Multiple Choice questions.

**Multiple Choice (10x1 = 10 points)**

1. What generally produces faster execution
   1. None of the below
   2. A purely interpreted language
   3. A hybrid language
   4. A purely compiled language
2. Select the correct order in the compilation process.
   1. Syntax Analysis then Lexical Analysis
   2. Code Generation, then Syntax, then lexical analysis
   3. Lexical Analysis, then Syntax Analysis, then Code Generation
   4. None of the above
3. In what language is Unix mostly written?
   1. ALOGOL
   2. Assembly
   3. ‘C’
   4. ‘C++’
4. Attribute grammars cannot capture and/or enforce some features in a language that BNF grammars cannot.
   1. TRUE
   2. FALSE
5. If the grammar for a language is ambiguous, then some valid expression in that language has more than one parse tree.
   1. TRUE
   2. FALSE
6. ALGOL 60 was defined using which meta-language. (Choose the best answer)
   1. BNF
   2. EBNF
   3. XML
   4. UML
7. What was wrong with using machine code. (Choose the best answer)
   1. Poor readability
   2. Poor modifiability
   3. a, b and d
   4. Expression coding was tedious
8. What are Predicate Functions?
   1. They are only applicable in human languages, not computer languages.
   2. They state the static semantic rules for a language and are associated with grammar rules.
   3. They state the grammar rules of a language and are associated with semantic rules.
   4. There is no such thing as a predicate function.
9. What is the difference between synthesized and inherited attributes?
   1. Inherited attributes depend on information from descendants in the parse tree.
   2. Inherited attributes are useful only in object-oriented languages.
   3. Synthesized attributes are not computed.
   4. Synthesized attributes depend on information from descendants in the parse tree.
10. Which is not true about Derivations?
    1. A derivation must be either leftmost or rightmost
    2. A sentence is a sentential form that has only terminal symbols
    3. A leftmost derivation is one in which the leftmost nonterminal in each sentential form is the one that is expanded
    4. Every string of symbols in a derivation is a sentential form

**Pseudocode (choose 1, 5 points)**

1. Pseudocodes were invented to solve many problems. What were two principle issues designers where trying to overcome?
   1. Poor readability
   2. Poor modifiability
   3. Expression coding was tedious
   4. Machine deficiencies--no indexing or floating point
2. **The Pseudocode is described by four principles. (automation, orthogonality, regularity, security) Give a description of 3 of these principles)**

Automation – Once a program starts it will handle the next tasks by itself without any human assistance. The code is automated.

Orthogonality – A set of primitive constructs can be combined in a small number of ways. It allows for simplicity in a programming language. It also means that operations end up changing one thing without changing anything else.

Regularity – The code must be reliable and routine. The code should be able to handle all inputs and give output without crashing. Should be able to handle any errors.

**Chapter 1 (10 points, choose 3)**

1. **Describe the four categories of programming languages. Give the name and a brief description**

Imperative – Include scripting, visual, and object-oriented programming languages. Deals with variables, assignment statements and iteration. Includes languages such as C, Java, C++ just to name a few.

Functional – Computations are made by applying functions to given parameters. Includes languages such as LISP, Scheme, F# to name a few.

Logic – Rules are heavily specified but not in any order. Prolog is a main language for this category.

Markup/Programming hybrid – Extensions for markup languages that support some programming as well. A couple languages in this category are JSTL and XSLT.

1. What does orthogonality mean?

Orthogonality

* 1. A relatively small set of primitive constructs can be combined in a relatively small number of ways
  2. Every possible combination is legal

1. **Name and describe 3 of the 5 programming domains listed in the book.**

* Scientific applications
  + Large numbers of floating point computations; use of arrays
  + Fortran
* Business applications
  + Produce reports, use decimal numbers and characters
  + COBOL
* Artificial intelligence
  + Symbols rather than numbers manipulated; use of linked lists
  + LISP
* Systems programming
  + Need efficiency because of continuous use
  + C

1. **There are 4 main criteria for language evaluation, choose and describe 2 of them.**

* **Readability**: the ease with which programs can be read and understood

Maintenance cost drove crossover from focus on machine to human.

* **Writability**: the ease with which a language can be used to create programs
* **Reliability**: conformance to specifications (i.e., performs to its specifications)
* **Cost**: the ultimate total cost

**Chapter 2 (5 points)**

1. **Algol 68 was special for several reasons. One can easily think of four, but I would like you describe 2 reasons why it was special. You can site reasons like implementation details, features, acceptance, etc. (but you need to describe them).**

One of the special features Algol 68 brought with it was the ability for arrays to be manipulated by the user, called flex arrays.

One other special feature that Algol 68 had was the ability for users to define data structures and declare variables before their first use.

**Chapter 3 (25 Points)**

1. Consider the grammar given below (5 points)

<pop> ::= [ <bop> , <pop> ] | <bop>

<bop> ::= <boop> | ( <bop> )

<boop> ::= x | y | z

1. What are the nonterminal symbols? (1 point)

<pop>, <bop>, and <boop>

1. What are the terminal symbols? (1 point)

x, y, z, (, )

(d) Draw a parse tree for the sentence (x). (3 points)

(note the parenthesis are part of the sentence)

<pop>

<bop>

( <bop> )

( <boop> )

(x)

1. (10 points) Show that the following grammar is ambiguous (Hint: first two levels

of both trees will be the first line)

<S> -> <A>

<A> -> <A> + <A> | <id>

<id> -> a | b | c

S S

A A

A + A A + A

A + A id id A + A

id id c a id id

a b b c

This grammar is ambiguous because we have two parse trees for the same string

1. (10 points) Using the following grammar, show the leftmost derivation for:

A = A \* (B + (C \* A)) (derivation, not the parse tree)

<assign> → <id> = <expr>

<id> → A | B | C

<expr> → <id> + <expr>

| <id> \* <expr>

| ( <expr> )

| <id>

<assign> → <id> = <expr>

→ A = <expr>

→ A = <id> \* <expr>

→ A = A \* <expr>

→ A = A \* ( <expr> )

→ A = A \* (<id> \* <expr>)

→ A = A \* (B + <expr>)

→ A = A \* (B + (<id> \* <expr>))

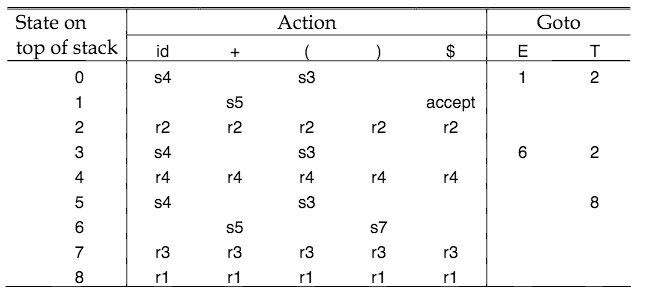
→ A = A \* (B + (C \* <expr>))

→ A = A \* (B + (C \* <id>))

→ A = A \* (B + (C \* A))

1. **Chapter 4 (10 Points)**

Consider the following state table:



Here is the associated grammar:

1) E –> E + T

2) E –> T

3) T –> (E)

4) T –> id

Show the parse of (id) + (id) using the LR algorithm (Hint: Shift-Reduce Algorithm) practiced in class.

Stack Input Action

$ (id) + (id)$ Shift 3

$( id) + (id)$ Shift 4

$(id ) + (id)$ Reduce 4 T→id

$(T ) + (id)$ Reduce 2 E→T

$(E ) + (id)$ Shift 7

$(E) + (id)$ Reduce 3 T→(E)

$T + (id)$ Reduce 2 E→T

$E +(id)$ Shift 5

$E + (id)$ Shift 3

$E + ( id)$ Shift 4

$E + (id )$ Reduce 4 T→id

$E + (T )$ Reduce 2 E→T

$E + (E )$ Shift 7

$E + (E) $ Reduce 3 T→(E)

$E + T $ Reduce 1 E→E + T

$E $ Accept

1. **C Programming (10 Points) – write a C program that has a main function and one other function. The program will accept three different positive integer numbers from the user, and print out the smallest number. The main function will handle the input and the output, the other function will just do the determination of the smallest number and return the results. Assume all three numbers are positive integers, and all three are different numbers (no edit checking required).**

**#include <stdio.h>**

**#include <stdlib.h>**

**// Function declaration**

**int smallestNum(int[]);**

**/\*\***

**\* Main function retrieves 3 numbers from user and calls smallestNum**

**\* to determine the smallest number entered and displays the result.**

**\*/**

**int main(int argc, char \*argv[]) {**

**int threeNums[3];**

**int number;**

**int i;**

**printf("Enter three postive numbers: \n");**

**while (i < 3) {**

**scanf("%d", &number);**

**if (number <= 0) {**

**printf("Please enter a postive number. \n");**

**} else {**

**threeNums[i] = number;**

**i++;**

**}**

**}**

**printf("The smallest number is %d", smallestNum(threeNums));**

**return 0;**

**}**

**/\*\***

**\* Function takes user input and returns the smallest number.**

**\*/**

**int smallestNum(int threeNums[]) {**

**int i;**

**int min = threeNums[0];**

**for (i = 1; i < 2; i++) {**

**if (threeNums[i] < min) {**

**min = threeNums[i];**

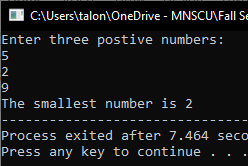
**}**

**i++;**

**}**

**return min;**

**}**

****