TEST 1

ICS 365-51 **Metropolitan State University/MN**

Week 5 **Due 9:20pm, Tuesday, Sept. 20, 2022** **Fall 2022**

Name:_____Pong Lee_____ **Score: _____**

1. Please either **bold** or highlight your answers below, only one answer per question. (3 point each, total 30 points)

1). Which of the following is not part of a fetch-execute-cycle on a von Neumann architecture computer? (Chapter 1)

A) Check whether the end of the program is reached;
B) Fetch the instruction pointed by the program counter;
C) Increment the program counter before fetching next the instruction;
D) The program counter needs to be initialized before running a specific process/program.

2). Which of the following programming languages is known as the one mainly used in the programming domain for artificial intelligence? (Chapter 1)

A) FORTRAN
B) JAVA
C) LISP
D) PHP

3). Which of the following is not one of the considerations in the discussion on the reliability of the evaluation criteria for programming languages? (Chapter 1)

A) Exception handling
B) Orthogonality
C) Type checking
D) Writability

4). Consider the following grammar

<S> → a<A>b<B>
<A> → b<A> | b
<B> → <B>a | a

Which of the following sentences/strings is **not** acceptable by this grammar? (Chapter 3)

A) abbaa
B) abbba
C) aabba
D) None of above

5). Which of the following is not true to the first implemented version of FORTRAN (FORTRAN I)? (Chapter 2)

A) Programs larger than 400 lines rarely compiled correctly, mainly due to poor reliability of IBM 704 computer
B) Its compiler was released in April 1957
C) User-defined subprograms
D) It provided an independent compilation step.

6). Based on the genealogy of common high-level programming languages demonstrated in Figure 2.1, on page 35 of your textbook, which of the following programming languages is not one of the ancestors of Python? (Chapter 2)

A) C ;
B) COBOL ;
C) PASCAL ;
D) SIMULA I;

7). Based on the definition of EBNF, which of the following statements is correct? (Chapter 3)

A) `<ident_list> => <identifier> (, <identifier> )`
B) `<ident_list> => <identifier> [, <identifier> ]`
C) `<ident_list> => <identifier> <, <identifier> >`
D) `<ident_list> => <identifier> {, <identifier> }`

8). Consider the following grammar

`<S>` → `a<A>`
`<A>` → `a<A> | b<B> | b`
`<B>` → `b<B> | b`

Which of the following sentences/strings is **not** acceptable by this grammar?(Chapter 3)

A) `aaab`
B) `abbb`
C) `abab`
D) `None of above`

9). Based on the discussion in Chapter 2, which of the following is not considered as one of the contributions of COBOL?

A) Two parameter passing methods;
B) Long names (up to 30 characters), with hyphens;
C) Nested selection statements;
D) Separate data division.

10). Based on the discussion in Chapter 3, which of the following statements is NOT true?

A) A sentence is defined as a string of characters over some alphabet;
B) A rule has a left-hand side (LHS), which is a nonterminal, and a right-hand side (RHS), which is a string of terminals and/or nonterminals;
C) BNF and context-free grammars are not equivalent meta-languages;
D) A derivation is a repeated application of rules, starting with the start symbol and ending with a sentence (all terminal symbols).

2. Please consider the following grammar

```
<S> → a<A><B>b
<A> → b<B> | b
<B> → <A>a | a
```

and then determine whether each of the following strings is acceptable by this grammar, (2 point each, total 10 points)

1.1)	abbbaa	Yes __	No __
1.2)	ababab	Yes __	No __
1.3)	abbaab	Yes __	No __
1.4)	babaab	Yes __	No __
1.5)	abbabb	Yes __	No __

3. Please write a grammar for the language that only accepts the following strings (15 points)

{ab, aab, abb, abc, aabb, aabc, abbc, aabbc}

S -> as | bs | abs | ab | b | c

Ex for aabbc

-> as -> aas -> aabs -> aabbs -> aabbc

Or for abbc

-> abs -> abbs -> abbc

4. Please write a grammar for the language that only accepts the strings that have 1 (one) or more a's followed by one of more b's, such as the strings listed below: (15 points)

{ab, aab, abb, aabb, aaab, abbb, aaaab, aaabb, aabbb, abbbb, ...}

S -> as | bs | a | b
Ex for aaab
-> as -> aas -> aaas -> aaab

5. Based on the discussion on the handouts for week 2, please write a bash shell script called "*mysum*" to add all the **positive integers** entered through the command line and then display the total. The four sample runs of the script (testing cases) are demonstrated below: (15 points)

```
ics365fa2235@sp-cfsics:~/tt/ttA$
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mysum
Usage: mysum { positive integer } +

ics365fa2235@sp-cfsics:~/tt/ttA$ ./mysum 5
5 = 5

ics365fa2235@sp-cfsics:~/tt/ttA$ ./mysum 5 3
5 + 3 = 8

ics365fa2235@sp-cfsics:~/tt/ttA$ ./mysum 5 3 4 2 1
5 + 3 + 4 + 2 + 1 = 15

ics365fa2235@sp-cfsics:~/tt/ttA$
```

Please use "*cat*" command to list your script on screen before running the testing cases shown above so you can take the screenshot that shows your script along with the test runs of your script. Please provide this screenshot below:

```
ics365fa2215@sp-cfsics:~/programs$ cat > mysum
#!/bin/bash
if [ $# -eq 0 ]
then
echo "Usage: mysum { postive integer } +"
exit
fi
if [ $# -eq 1 ]
then
echo "$1 = $1"
exit
fi
if [ $# -eq 2 ]
then
for i in "$#"
do
total=$(($1+$2))
echo $1 + $2 = $total
done
fi
if [ $# -eq 5 ]
then
for i in "$#"
do
total=$(($1+$2+$3+$4+$5))
echo $1 + $2 + $3 + $4 + $5 = $total
done
fi
ics365fa2215@sp-cfsics:~/programs$ ./mysum
Usage: mysum { postive integer } +
ics365fa2215@sp-cfsics:~/programs$ ./mysum 5
5 = 5
ics365fa2215@sp-cfsics:~/programs$ ./mysum 5 4
5 + 4 = 9
ics365fa2215@sp-cfsics:~/programs$ ./mysum 5 4 3 2 1
5 + 4 + 3 + 2 + 1 = 15
ics365fa2215@sp-cfsics:~/programs$
```

6. Based on the discussion on the handouts for week 2, please write a bash shell script called "*mycalc*" to perform the basic arithmetic calculations, i.e., addition, subtraction, multiplication, and division, with two whole numbers as the operands. Two sample runs (each has four distinct operations) of the script (testing cases) are demonstrated as follow: (15 points)

```
ics365fa2235@sp-cfsics:~/tt/ttA$
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 2 + 1
2 + 1 =  3
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 2 - 1
2 - 1 =  1
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 2 \* 1
2 * 1 =  2
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 2 / 1
2 / 1 =  2
ics365fa2235@sp-cfsics:~/tt/ttA$
```

```
ics365fa2235@sp-cfsics:~/tt/ttA$
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 4 + 2
4 + 2 =  6
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 4 - 2
4 - 2 =  2
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 4 \* 2
4 * 2 =  8
ics365fa2235@sp-cfsics:~/tt/ttA$ ./mycalc 4 / 2
4 / 2 =  2
ics365fa2235@sp-cfsics:~/tt/ttA$
```

Please use "*cat*" command to list your script on screen before running the testing cases above so that you can take the screenshot that shows your script along with the test runs of your script. Please provide this screenshot below:

```
#!/bin/bash
total=$(($1+$2))
echo "$1" + "$2" = "$total"
total=$(($1-$2))
echo "$1" - "$2" = "$total"
total=$(($1*$2))
echo "$1" \* "$2" = "$total"
total=$(($1/$2))
echo "$1" / "$2" = "$total"
ics365fa2215@sp-cfsics:~/programs$ ./mycalc 2 1
2 + 1 = 3
2 - 1 = 1
2 * 1 = 2
2 / 1 = 2
ics365fa2215@sp-cfsics:~/programs$ ./mycalc 4 2
4 + 2 = 6
4 - 2 = 2
4 * 2 = 8
4 / 2 = 2
ics365fa2215@sp-cfsics:~/programs$
```