**Question 1**

**2 / 2 pts**

This expression... matches this...

                (.a)+b             ra5afab

**Correct!**

**True**

False

**Question 2**

**2 / 2 pts**

This expression... matches this...

       [aeiou][0-9]            aei6

**True**

**Correct!**

False

**Question 3**

**2 / 2 pts**

Prolog is based on the lambda calculus concept to develop programs.

True

**Correct!**

**False**

**Question 4**

**0 / 2 pts**

The following two regular expressions are the same.  In other words, they denote the same language.

           (a | b)+(a | b)\*b+b\*

           (a | b)+b\*

(where + means one or more.)

**You Answered**

True

**Correct Answer**

**False**

**Question 5**

**2 / 2 pts**

CLISP is case-sensitive. e.g.  'format' must be in lower case in the following statement.

(format t "Hello world~%")

True

**Correct!**

**False**

**Question 6**

**2 / 2 pts**

Swift is a replacement for C#.

True

**Correct!**

**False**

**Question 7**

**2 / 2 pts**

The following two regular expressions are the same.  In other words, they denote the same language.

               a+a\*a\*b\*b+b\*

               a+b+

(where + means one or more.)

**Correct!**

**True**

False

**Question 8**

**2 / 2 pts**

COBOL is mainly designed for scientific applications.

True

**Correct!**

**False**

**Question 9**

**2 / 2 pts**

Imperative language is also known as declarative language.

True

**Correct!**

**False**

**Question 10**

**2 / 2 pts**

C was designed for systems programming (at Bell Labs by Dennis Richie) in 1972.

**Correct!**

**True**

False

**Question 11**

**2 / 2 pts**

 List the reasons why a programming language gets popular (listed in the lecture ppt files) - multiple answers.

**Correct!**

**1. Killer App**

**Correct!**

**2. Platform Exclusivity**

3. Rich in features

**Correct!**

**4. Quick Upgrade**

5. Developed by NASA

**Question 12**

**2 / 2 pts**

**Select all the Language Evaluation Criteria - multiple answers:**

**Correct!**

**1. Readability**

**Correct!**

**2. Writability**

3. Fast Compilation

**Correct!**

**4. Cost**

**Correct!**

**5. Reliability**

**Question 13**

**2 / 2 pts**

Which one(s) of the following strings will match this regular expression?

        [^acd]bk\*g

1. aebkkkg

2. cbkkg

3. ebbkkkg

**Correct!**

**4. ebg**

**Question 14**

**2 / 2 pts**

Which one(s) of the following is/are NOT imperative language?

C

Perl

JavaScript

**Correct!**

**Prolog**

**Question 15**

**2 / 2 pts**

**List the reasons for Studying Concepts of Programming Languages - multiple answers:**

**Correct!**

**1. Increased ability to express ideas**

**Correct!**

**2. Improved background for choosing appropriate languages**

**Correct!**

**3. Increased ability to learn new languages**

4. Better understanding of a stake holder

**Correct!**

**5. Better use of languages that are already known**

**Correct!**

**6. Better understanding of significance of implementation**

**Question 16**

**2 / 2 pts**

Select all the language **Implementation Methods - multiple answers:**

**Correct!**

**1. Compilation**

2. Heap Interpretation

**Correct!**

**3. Hybrid Implementation**

**Correct!**

**4. Pure Interpretation**

**Question 17**

**2 / 2 pts**

Which one of the following strings can be generated by this formal grammar?   
            S → aSb  
            S → ε

1. ababab

2. aabbaabb

**Correct!**

**3. aaabbb**

4. abba

**Question 18**

**0 / 2 pts**

Which one(s) of the following regular expressions will match this DFA?

**You Answered**

1. a\* | b\*

2. a\*b | b\*a

**Correct!**

**3. a+ | b+**

4. ab+ | ab\*

**Question 19**

**2 / 2 pts**

Type-0 grammars in Chomsky hierarchy are:

1 Restricted grammar

2. Context-sensitive grammar

3. Linear bounded grammar

**Correct!**

**4. Unrestricted grammar**

**Question 20**

**2 / 2 pts**

Select all the **Programming Domains mentioned in the class lecture - multiple answers:**

**Correct!**

**1. Scientific applications**

**Correct!**

**2. Business applications**

**Correct!**

**3. Artificial intelligence (AI)**

4. Linear bounded automaton

**Correct!**

**5. Web Software**

**Correct!**

**6. Systems programming**

**Question 21**

**3 / 3 pts**

Will the following print true or false?

System.out.println(Pattern.matches("[789][0-9]{4}", "9956301"));

**Correct!**

**Correct Answers**

**False**

false

**Question 22**

**3 / 3 pts**

The corresponding automaton for Type – 0 language is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Correct!**

**Correct Answers**

Turing machine

**Question 23**

**3 / 3 pts**

The following CLISP statement will print \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

(format t "Items = ~a ~%" (cddr '(superman batman aquaman flash joker)))

Your Answer:

**Items = (AQUAMAN FLASH JOKER)**

**Correct Answer**

**Question 24**

**3 / 3 pts**

COBOL stands for ?

**Correct!**

**Correct Answers**

common business-oriented language

common business oriented language

**Question 25**

**3 / 3 pts**

Chomsky's type-1 grammars generate context-sensitive languages. These grammars have rules with the form

                       α A β  -> α ɣ β

with A a nonterminal and α, β, and ɣ  strings of terminals and/or nonterminals.

On the LHS of the rule:  α A β -> α ɣ β,

      1. α is the  for the context-sensitive grammar.

      2. β is the  for the context-sensitive grammar.

**Answer 1:**

**Correct!**left context

**Correct Answer**

left context

**Answer 2:**

**Correct!**right context

**Correct Answer**

right context

**Question 26**

**3 / 3 pts**

Given the lambda expression in this format \(head).(body) where ‘\’ = lamda (λ).

if we have this lambda expression

                 (\x.xy) c

what will be the result?

**Correct!**

**Correct Answers**

cy

**Question 27**

**3 / 3 pts**

The corresponding automaton for regular expression is   \_\_\_\_\_\_\_\_\_\_\_\_

**You Answered**

**Correct Answers**

Finite State automaton

Finite State Machine

**Question 28**

**3 / 3 pts**

A formal grammar of the **Chomsky hierarchy** consists of a finite set of production rules (left-hand side (LHS) → right-hand side (RHS)), where each side consists of a finite sequence of a start symbol and these 2 symbols.

Please list these 2 symbols :

 ,

 ,

**Answer 1:**

**You Answered**terminal

**Correct Answer**

a finite set of nonterminal symbols

**Correct Answer**

Nonterminals

**Correct Answer**

Non-terminals

**Answer 2:**

**You Answered**nonterminal

**Correct Answer**

a finite set of terminal symbols

**Correct Answer**

terminals

**Question 29**

**4 / 4 pts**

Write the corresponding regular expression for the following finite-state automaton.

**Correct!**

**Correct Answers**

a+[b|c]d\*a[d|e]

aa\*[b|c]d\*a[d|e]

**Question 30**

**4 / 4 pts**

Given the lambda expression in this format \(head).(body) where ‘\’ = lambda (λ).

if we have this lambda expression

                 (\x.xy)(\a.aa)

what will be the result?

**Correct!**

**Correct Answers**

yy

**Question 31**

**4 / 4 pts**

Describe what is an ambiguous grammar and give a common example to demonstrate this problem.

Your Answer:

Ambiguous grammar is when more than one distinct derivations of a string exists resulting in differing parse trees. A common example of this problem is if-then-else statement grammar productions.

**Question 32**

**4 / 4 pts**

Rewrite the following BNF grammar to EBNF:  
      <id> -> <letter>  
                   |  <id><letter>  
                   |  <id><digit>

Your Answer:

<id> -> <letter> {<letter>  | <digit>}

**Question 33**

**1 / 5 pts**

Given the following grammar:

E  -> T + T | T  
T ->  T \* F | F  
F ->  i  
Input string is: i+i

Please show how the Shift Reduce operations are done.

You can copy the following table to your text box and fill in the information.  Or you can create your own table there.

|  |  |  |
| --- | --- | --- |
| Stack | Input | Action |
| $ | i+i$ | Shift |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Your Answer:

|  |  |  |
| --- | --- | --- |
| Stack | Input | Action |
| $ | i+i$ | Shift |
| $i | + | Shift |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Question 34**

**3 / 5 pts**

Convert the following regular expression to NFA/DFA.

      (a|b)\* (ba\*b|ab\*a)

You can draw your NFA/DFA (picture) and upload it here.

[DFA.pdf](https://smu.instructure.com/files/4544743/download)

**Question 35**

**5 / 5 pts**

Given the following DFA, write the corresponding code in C or Java or pseudo code to accept or reject a string.  The output is 'accept' or 'reject'...

Your Answer:

string function1(){

     string str = “”;

      c = NextChar();

if (c == ‘f’) {

      str = str + c;

      c = NextChar();

while (c == ‘r’){

     str = str + c;

      c = NextChar();

}

if (c == ‘e’) {

     str = str + c;

     c = NextChar();

}

if (c == ‘e’) {

     str = str + c;

     c = NextChar();

     return str;

}

return error;

}

**Question 36**

**4 / 5 pts**

Given the following grammar:

<expr> -> <term>   
         | <term> + <term>   
<term> -> <factor>   
         | <factor \* <factor>    
<factor> -> digit | (<expr>)

Write a recursive descent parser in C or Java or a language of your choice - high level code that covers the main logic.  No need to code the whole program if you do not have enough time.

Your Answer:

expression:

   term();

    if(token == "+"):

              expression = term() + term()

   term:

       factor()

           if(token =="\*"):

                   term = factor() \* factor()

factor:

           digit = constant;

              return digit;