**I The theory part(inc essay and seminar)**

1. **(x points)What language(s) can be described by following chracateristics? Explain.**

* data types, data objects, and procedure specifications can be encapsulated into a package. This supports the program design of data abstraction.
* has very good exception handling capabilities which allow the program to handle its own run-time errors.
* it is possible to write a procedure (for example, a sorting procedure) which does not require a data type to be specified.
* supports parallel and concurrent execution of tasks.
* supports object-oriented programming
* more flexible libraries
* better control mechanisms for shared data

1. **(x points)What language(s) can be describes by following characteristics? Explain.**

* is completely free form and has no reserved keywords.
* precisely defines its data types without regard for any hardware.
* is a block-oriented language, consisting of packages, begin blocks, and statements. This type of structure allows the programmer to produce highly-modular applications.
* contains control stuctures.
* supports arrays, structures, unions, arrays of structures or unions, structures or unions of arrays, and combinations thereof.
* provides four different storage classes: AUTOMATIC, STATIC, CONTROLLED, and BASED. Application objects' data type, representation, nature of use, etc decides the type of storage class used for each.

1. **(x points)Explain the "dangling else" problem and how you would solve it.**

**(x points)What are the possibilities for implicit variable declaration in various languages? Name a few examples.**

1. **(x points)What are the different types of subroutines in various languages? Name a few examples.**
2. **(x points)What is the difference between ANTLR4 and Bison grammar parsing? ( LL(\*) vs LALR )**
3. **(x points)What is abstract syntax tree and where is it used?**

**II The practice part(inc home task)**

1. **(x points)Create syntax diagram or description in a metalanguage for the following sentences:**

IF X(J)>X(J+1) THEN DO;

Y = X(J);

X(J) = X(J+1);

X(J+1) = Y;

END;

1. **(x points)Create syntax diagram or description in a metalanguage for the following sentences:**

for i := 1 to n do

p := p \* p;

repeat

sum := sum + a[k];

k := k + 1

until k = m;

1. **(x points)Create syntax diagram or description in a metalanguage for the following sentences:**

select

when (condition = 0) do;

(sentence A)

end;

when (condition = 1) do;

(sentence B)

end;

otherwise do;

(sentence C)

end;

end;

1. **(x points)Create syntax diagram or description in a metalanguage for the following sentences:**

assign(it, 'somefile.txt');

reset(it);

readln(it, s);

close(it);

println(s);

1. **(x points)Translate to postfix form and assign the order of operations:**

Y = A / S - K \* ( P / M + V) + L \* I

1. **(x points)Translate to infix form and add parantheses where necessary:**

Y = A T K \* + F P R Q / - \* I + C G \* +

1. **(x points)Translate given sentences from C to pseudo-C using three main sentences:**

for(i = 0; i < n; i++)

{ s = s + a[i];}

1. **(x points)Comment and optimize code:**

S=1

DO 1 I = 1,N

S=S\*(-1)\*\*I\*J

1 CONTINUE

1. **(x points)Write scanner rules for parsing integer values, floating point values and variables.**
2. **(x points)Write scanner and/or parser rules for parsing following code:**

( 3 8 5 7 11 6 )

( 1 )

( 3 5 7 )

( )

1. **(x points)Write grammar rules that would accept the following HTML code**

<table>

<tr>

<th>some fancy text</th>

<th>more fancy text</th>

</tr>

<tr>

<td>some mundane text</td>

<td>

<table>

<tr>

<th>fancy words</th>

<td>nonfancy words</td>

</tr>

<tr>

<th>more fancy words</th>

<td>more nonfancy words</td>

</tr>

</table>

</td>

</tr>

</table>

1. **(x points)We have following Flex scanner rules:**

digit [0-9]

alpha [a-zA-Z]

alphanum [0-9a-zA-Z]

%%

"PROCEDURE" { return PROC; }

"PRINT" { return PRINT;}

{digit}+ { return INT;}

{alpha}{alphanum}\* { return VAR;}

[=;{}()+-/\*] { return yytext[0];}

[ \t\n] ; // do nothing

**and following source code:**

PROCEDURE Calculate ( x )

{ y = x \* x - 5; PRINT y;}

**What would be the sequence of tokens that Flex generates from this code?**

1. **(x points)We have following source code:**

BEGIN

a = 5 + 2;

b = 10 - 2;

IF a > b THEN c = a;

ELSE c = b;

ENDIF;

END;

**Define tokens and write parser rules (in Bison or equivalent) for parsing these sentences.**

1. **(x points)Translate the following code to C:**

BUB:   PROCEDURE(ARRAY,N);

         DECLARE (I,J) FIXED BIN(15);

         DECLARE S BIT(1);

         DECLARE Y FIXED BIN(15);

         DO I = N-1 BY -1 TO 1;

           S = '1'B;

           DO J = 1 TO I;

             IF X(J)>X(J+1) THEN DO;

               S = '0'B;

               Y = X(J);

               X(J) = X(J+1);

               X(J+1) = Y;

               END;

             END;

           IF S THEN RETURN;

           END;

         RETURN;

1. **(x points) \*\*ANTLR4 grammar is given:**

grammar Expr;

prog : stat+ ;

stat : expr NEWLINE

| ID '=' expr NEWLINE

| NEWLINE

;

expr : expr ('\*'|'/') expr

| expr ('+'|'-') expr

| INT

| ID

| '(' expr ')'

;

ID   : [a-zA-Z]+ ;

INT : [0-9]+ ;

NEWLINE : '\r'? '\n' ;

WS   : [ \t]+ -> skip ;

**Which inputs are recognized as correct according to the grammar?**

a = 1 + 2

a2 = 2 - 4

(b + 3);

c = a / ( 1 + b\*(2-c))

d = 4 / 0.5

5 + 4 = e

f = a2

g = a        \*  2+        n

1. **(x points)\*\*ANTLR4 grammar is given:**

grammar Expr;

prog : stat+ ;

stat : expr NEWLINE

| ID '=' expr NEWLINE

| NEWLINE

;

expr : expr ('\*'|'/') expr

| expr ('+'|'-') expr

| FLOAT

| ID

| '(' expr ')'

;

ID   : [a-zA-Z][a-zA-Z0-9\_]+ ;

FLOAT : [0-9]+'.'[0-9]\*

| [0-9]\*'.'[0-9]+

| [0-9]+

;

NEWLINE : ';';

WS   : [ \t\r\n]+ -> skip ;

**Which inputs are recognized as correct according to the grammar?**

a = .5 + 2.;

a2 = 2 - 4;

(2b + 3);

c = a / ( 1 + b\*(2-c\_b));

d = 4 / 0.5

5 + 4 = e;

f = a 2;

g = a        \*  2+        n;