Name________ANSWER KEY________________________

CS471, Programming Language Structure Spring, 2002

Examination 1

Closed Book. Answer all the questions. The total points for each question or part of a question follows it in parentheses, thus: *(12)*

1

Below is a simplifed BNF grammar for the Conceptual Graph Interchange Format. <Label> is any string of printing characters and does not need a defining rule.

a) Does the sentence (POS[CAT*x1][MAT)(AGT[SIT]?x1) conform to the grammar? If not, why not? *(6)*

b) For every 'bound' label (e.g. ?x1) there should be a corresponding 'def' label. Can this kind of constraint be expressed in a BNF grammar? *(4)*

c) Rewrite the grammar in EBNF, removing as much recursion as possible within each rule, and using grouping and optional elements where possible. *(10)*

```
<CG>  ::= <Node> | <Node> <CG>
<Node> ::= <Relation> | <Concept>
<Relation> ::= ( <Label> <Arcs> )
<Arcs> ::= <Arc> | <Arc> <Arcs>
<Arc> ::= <Concept> | <BoundLabel>
<Concept> ::= [ <Label> ] | [ <Label> <DefLabel> ] | [ ]
<BoundLabel> ::= ? <Label>
<DefLabel> ::= * <Label>
```

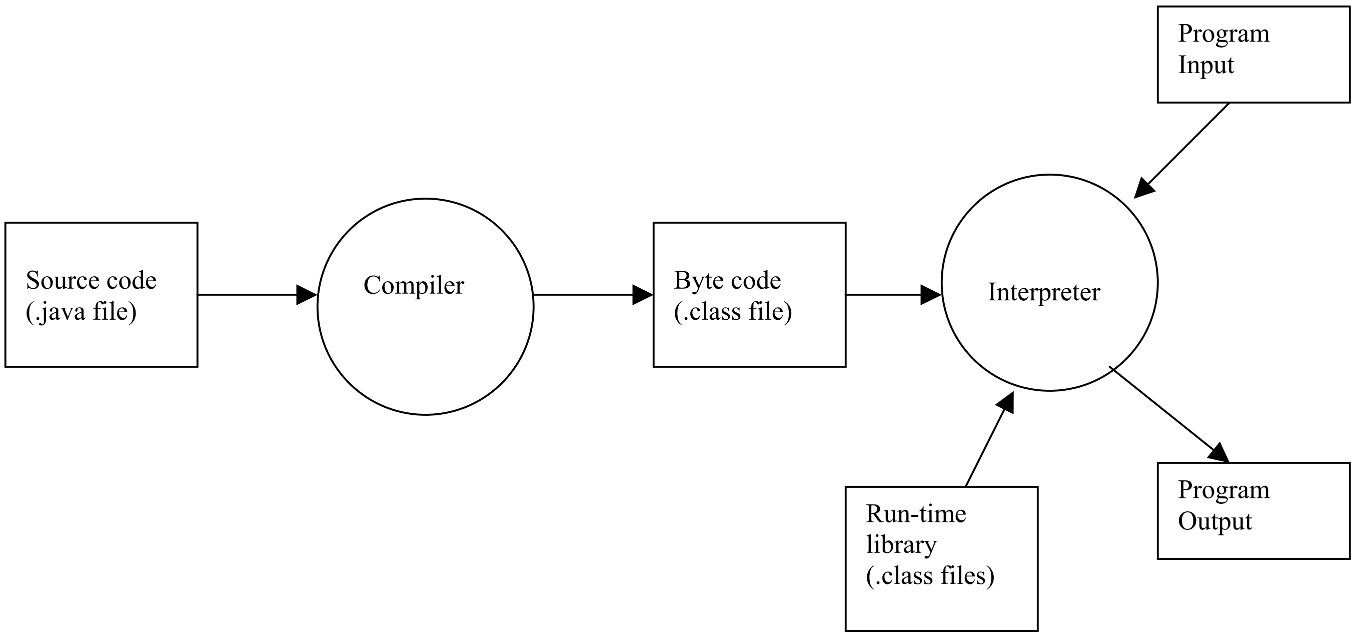
a) **No. Square brackets must be matched (rule for <Concept>) so [MAT) is impossible.**

b) **No. This is context sensitive. BNF can only represent context free languages.**

c) **CG ::= { Node }^{1}**
Node ::= Relation | Concept
Relation ::= '(' Label { Arc }^{1} ')'
Arc ::= Concept | BoundLabel
Concept ::= '[' [Label [DefLabel]] ']'
BoundLabel ::= '?' Label
DefLabel ::= '*' Label

2

Draw a diagram showing the various stages in the execution of a Java program. *(12)*



3

Below is a C++ program that uses reference parameters.

a) What is the value of x after the call to function B in the main function? *(10)*

b) Rewrite the code to use explicit pointers instead of references but leaving the meaning of the program unchanged (i.e. the value of x is unchanged at the end of the program.) *(6)*

```
int x;

void A(int& x){
  x = 1;
}

void B(int& y) {
  A(y);
}

main() {
  x = 2;
  B(x);
/* what is the value of x here? */
}
```

a) **The value of x is 1. [The two parameters x (in A) and y (in B) are aliases for the global variable x.**

b)

```
int x;

void A(int* x){
  *x = 1;
}

void B(int* y) {
  A(y);
}

main() {
  x = 2;
  B(&x);
/* what is the value of x here? */
}
```

4

Pascal has a type of loop that repeats its body until a certain condition holds. An example is:

```
repeat
  x := x * 2
until x > 1000;
```

which will repeatedly double the value of x until it exceeds 1000.

a) Write a suitable form for the abstract syntax of the repeat loop. *(6)* You can assume that (part of) the rest of the definition is:

```
S ::= I := E | S1 ; S2 | if B then S1 else S2 | while B do S
```

b) Write the operational semantics of both the while loop *and* the repeat loop using only a conditional form and recursion in a simple functional language such as the one given in class. Be careful to show the store as a parameter in your functions. *(8)*

c) Rewrite the operational semantics of the repeat loop using the semantic definition of the while loop you gave in b. This new one should be a non-recursive definition. *(6)*

a)

```
S ::= … | repeat S until B
```

b)

```
stmt(while B do S, s) = if bool(B,s) is true then
                             stmt(S;while B do S, s)
  else s

stmt(repeat S until B, s) =
                    if bool(B, s′) is true then s′
                    else stmt(repeat S until B, s′)
                        where s′ = stmt(S, s)
```

c)

```
stmt(repeat S until B, s) = stmt(S;while not B do S, s)
```

5

a) Why does the sequence of assignments:

```
x := 1.234;
x := 'a string';
```

make no sense in Pascal, but the corresponding sequence:

```
x <- 1.234.
x <- 'a string'.
```

makes perfect sense in Smalltalk? *(10)*

b) Explain why the sequence:

```
x = 1.234;
x = 1234;
```

does make sense in C. *(6)*

a) **Pascal is strongly typed and all variables must be declared with a static type that cannot change. The compiler will stop assignments of values of different types to the same variable.**

b) **C is a wekly typed language and all numeric types can be coerced to any other declared type. In the example the types double (all FP constants are automatically double) and integer can be coerced one to another.**

6

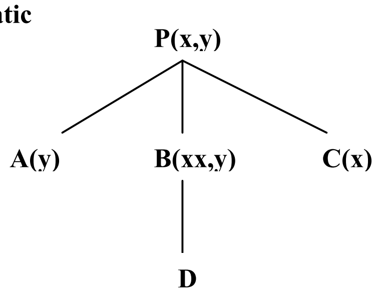
Below is a program written in Pascal syntax. The parameters are all pass-by-value. What does the program print:

a) In the case of static scoping? *(8)*

b) In the case of dynamic scoping? *(8)*

```
program P;
  var x : integer;
      y : integer;
  procedure A(y : integer);
  begin
    print(x);
    print(y)
  end;
  procedure B(xx : integer);
    var y : integer;
    procedure D;
    begin
      A(y)
    end;
  begin
    y := 4;
    D
  end;
  procedure C;
    var x : integer;
  begin
    x := 3;
    B(x)
  End;
begin
  x := 1;
  y := 2;
  C
end.
```

a) Static



P		A	B		C	D
x	y	y	xx	y	x	
1	2	4	3	4	3	

Prints P's x : 1
A's y : 4

b) P calls C calls B calls D calls A

Prints C's x : 3
A's y : 4