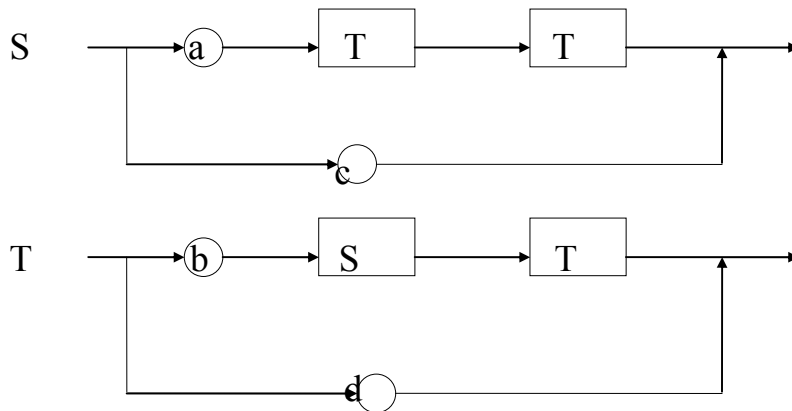


**Question 1.** [20 marks] A syntax chart is given as follows:



(1) Write a recursive descent parser for this chart in C, Pascal, or Java. Pseudo code notation is allowed, similar to the notes. 10 marks

(2) The trace for the string add! where ! is an end marker is given as follows:

History	Input string	Comment
S	add!	Consume a
S	dd!	Enter T
ST	dd!	Consume d
ST	d!	Exit from T
S	d!	Enter T
ST	d!	Consume d
ST	!	Exit from T
S	!	Exit from S
empty	!	Accept

Following this example, trace your parser with the string abcdbcd! 10 marks

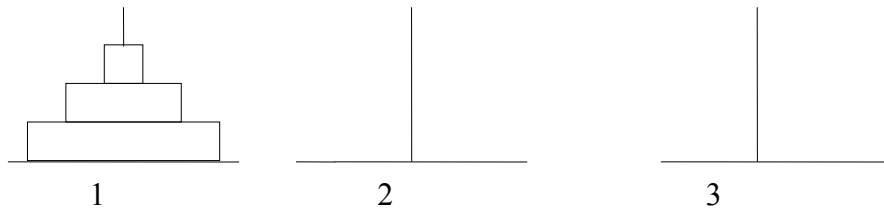
**Question 2.** The following is a PL/0 program with parameters for Tower of Hanoi.

```

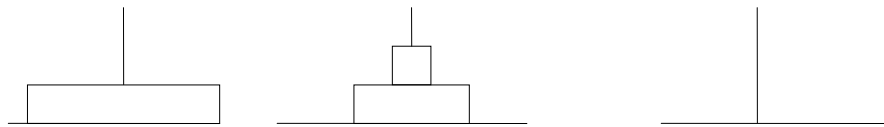
var x,y,z,n;
procedure hanoi(x,y,z,n);
begin
  if n>0 then begin
    call hanoi(x,z,y,n-1);
    write(x); write(z); write(999);
    call hanoi(y,x,z,n-1);
  end;
end;
begin
  x:=1; y:=2; z:=3; n:=3;
  call hanoi(x,y,z,n);
end.

```

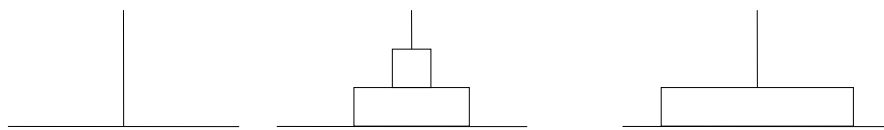
To move  $n$  disks from pole 1 to 3,



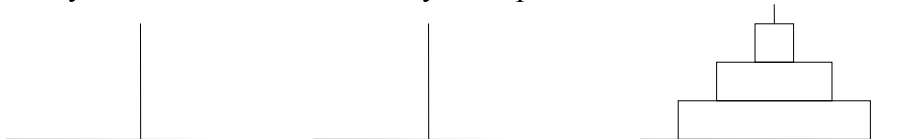
move  $n-1$  disks from pole 1 to 2 recursively,



then move a disk from pole 1 to pole 3,



and finally move  $n-1$  disks recursively from pole 2 to 3



The following is a PL0 program without parameters. It simulates parameter passing. The set of variables  $(x,y,z,n)$  plays the role of actual parameters and that of  $(xx,yy,zz,nn)$  plays the role of formal parameters. Write statements are to show movements of disks. The value 999 is to separate outputs.

```

var x,y,z,n;
procedure hanoi;
var xx,yy,zz,nn;
begin
  xx:=x; yy:=y; zz:=z; nn:=n;
  if nn>0 then begin
    y:=zz; z:=yy; n:=nn-1;
    call hanoi;
    write(xx); write(zz); write(999);
    x:=yy; y:=xx; z:=zz;
    call hanoi;
  end;
  x:=xx; y:=yy; z:=zz; n:=nn;
end;
begin
  x:=1; y:=2; z:=3; n:=3;
  call hanoi;
end.

```

The object code is given as follows:

0 jmp 0 46  
1 jmp 0 2  
2 inc 0 7  
3 lod 1 3  
4 sto 0 3  
5 lod 1 4  
6 sto 0 4  
7 lod 1 5  
8 sto 0 5  
9 lod 1 6  
10 sto 0 6  
11 lod 0 6  
12 lit 0 0  
13 opr 0 12  
14 jpc 0 37  
15 lod 0 5  
16 sto 1 4  
17 lod 0 4  
18 sto 1 5  
19 lod 0 6  
20 lit 0 1  
21 opr 0 3  
22 sto 1 6  
23 cal 1 2  
24 lod 0 3  
25 wrt 0 0  
26 lod 0 5  
27 wrt 0 0  
28 lit 0 999  
29 wrt 0 0  
30 lod 0 4  
31 sto 1 3  
32 lod 0 3  
33 sto 1 4  
34 lod 0 5  
35 sto 1 5  
36 cal 1 2  
37 lod 0 3  
38 sto 1 3  
39 lod 0 4  
40 sto 1 4  
41 lod 0 5  
42 sto 1 5  
43 lod 0 6  
44 sto 1 6  
45 opr 0 0  
46 inc 0 7  
47 lit 0 1  
48 sto 0 3

Beginning of main program. Secure space for (SL,DL,RA,x,y,z,n)  
load literal1  
store to x

```

49 lit 0 2    load literal2
50 sto 0 4    store to y
51 lit 0 3    load literal 3
52 sto 0 5    store to z
53 lit 0 3    load literal 3
54 sto 0 6    store to n
55 cal 0 2    call hanoi
56 opr 0 0    Return to operating system

```

start PL/0

```

1
3
999 /** move 1 to 3 **/
1
2
999 /** move 1 to 2 **/
3
2
999 /** move 3 to 2 **/
1
3
999 /** move 1 to 3 **/
2
1
999 /** move 2 to 1 **/
2
3
999 /** move 2 to 3 **/
1
3
999 /** move 1 to 3 **/
END PL/0

```

(A) Copy this code from address 2 to 45 onto the answer sheet and give a comment to each instruction describing the meaning in the source program. Follow the example starting from address 46. 10 marks

(B) After procedure hanoi is entered twice and the machine instruction at address 22 is executed the stack becomes as follows:

0 0 0 1 2 3 1 1 1 56 1 2 3 3 1 8 24 1 3 2 2

Explain the meaning of each element. 5 marks

(C) Show the contents of the stack after you execute instruction at 22 for the third time. 5 marks

Note. The program with parameters can handle parameters behind the scenes and much easier to understand.