Question 1. [15 marks for the whole question] A syntax chart for generating parenthesis strings is given as follows:

We define X as the starting symbol. The symbol “!” is an end marker explicitly used in the syntax chart.
(a) [5 marks] Show selection sets at branches (1) and (2).
(b) [5 marks] Write a recursive descent parser for this chart in C, Pascal, or Java. Pseudo code notation is allowed, similar to the one in the notes.
(c) [5 marks] The trace for the string ()! is given as follows:

<table>
<thead>
<tr>
<th>History</th>
<th>put string</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>()!</td>
<td>Enter S</td>
</tr>
<tr>
<td>XS</td>
<td>()!</td>
<td>Consume (</td>
</tr>
<tr>
<td>XS</td>
<td>)!</td>
<td>Enter S</td>
</tr>
<tr>
<td>XSS</td>
<td>)!</td>
<td>Exit from S</td>
</tr>
<tr>
<td>XS</td>
<td>)!</td>
<td>Consume (</td>
</tr>
<tr>
<td>XS</td>
<td>!</td>
<td>Enter S</td>
</tr>
<tr>
<td>XSS</td>
<td>!</td>
<td>Exit from S</td>
</tr>
<tr>
<td>XS</td>
<td>!</td>
<td>Exit from S</td>
</tr>
<tr>
<td>X</td>
<td>!</td>
<td>Consume !</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Exit from X</td>
</tr>
<tr>
<td>empty</td>
<td></td>
<td>Accept</td>
</tr>
</tbody>
</table>

Following this example, trace your parser with the string (())!.

Question 2. [25 marks for the whole question] The following is a PL/0 program with parameters for multiplication of two integers a and b, and its object code.

```pl0
var a, b, w;
procedure mult(x, y, var z) /** comment x, y: call by value **/
var w; /** comment z: call by address **/
begin
if x>0 then begin
    call mult(x-1, y, &w); /** comment w=(x-1)*y **/
w:=w+y; /** comment w=x*y **/
end;
z:=w;
end;
begin
a:=3; b:=5;
call mult(a, b, &w); /** comment w=a*b **/
write(w);
end.
```
15  /** comment This is the output given by the above “wrt” **/

(a) [10 marks] Copy this object code onto your answer book, and give a comment to each machine instruction, explaining the meaning in the source program.

(b) [10 marks] When the procedure is entered three times, and the instruction at 17 is executed for the first time, the stack looks like as follows:

```
0 0 0 0 3 5 0 1 1 35 3 5 6 0 1 7 16 2 5 13 0 1 14 16 1 5 20 0 0
```

Explain the meaning of each element. Undefined values are initialised to 0.

(c) [5 marks] Give the snapshot of the stack after the instruction at 17 is executed for the third time.