Print your name neatly in the space provided below.

Name: 
ID Number: 

Grade: 

<table>
<thead>
<tr>
<th>Question</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (10 marks): Consider the following grammar where \{ A, B, C, S \} is the set of non-terminals, \{ x, y, z, EOI \} is the set of terminals, and S is the start symbol.

\[
S := ABC \\
A := xy \mid \varepsilon \\
B := y \mid \varepsilon \\
C := z \mid \varepsilon
\]

a) Compute the FIRST set of all non-terminals.
b) Compute the FOLLOW set of all non-terminals.
Problem 2 (10 marks): The human gene sequence can be modeled as a string. According to Durbin et al in “Biological Sequence Analysis”, “The human FMR-1 gene sequence contains a triplet repeat region in which the sequence CGG or AGG is repeated a number of times. The number of triplets is highly variable between individuals, and increased copy number is associated with fragile X syndrome, a genetic disease that causes mental retardation and other symptoms in one out of 2000 children”.

The pattern is bracket by GCG and CTG, so we get the regular expression

GCG (CGG | AGG)* CTG.

c) Construct a NFA to recognize the regular expression. Draw the NFA diagram, clearly mark the start and accepting state.
d) Construct a DFA to recognize the regular expression. Draw the DFA diagram, clearly mark the start and accepting state.
Problem 3 (5 marks): Write a regular expression to match all strings that contain exactly five vowels and the vowels are in alphabetical order.
Problem 4 (10 marks): Convert the following NFA to a DFA. Clearly mark each DFA state with the corresponding NFA states.
**Problem 5 (15 marks):** Javascript Object Notation (JSON) is a subset of JavaScript which deals with describing constant objects. Given its popularity, it has become a popular mechanism for data exchange among distributed web services.

An example of JSON expression is as follows:

```
[ { "foo" : 12,  "bar" : "boom" }, 12, 13, “name” ]
```

The grammar of JSON can be described as follows:

01 object := {} | { members }  
02 members := pair | members , pair  
03 pair := string : value  
04 array := [] | [ elements ]  
05 elements := value | elements , value  
06 value := string | number | object | array | true | false | null

e) Modify the grammar so that it is suitable for top-down recursive descent parsing. Explain the reason why the modification is necessary.
f) Assume the lexical analysis function `yylex()` is available, which return the following token code:

```c
typedef enum {
    TOKEN_STRING = 0,
    TOKEN_NUMBER = 1,
    TOKEN_TRUE = 2,
    TOKEN_FALSE = 3,
    TOKEN_LOBJECT    = '{',
    TOKEN_ROBJECT  = '}',
    TOKEN_LLIST = '[',
    TOKEN_RLIST = ']',
    TOKEN_COMMA = ',',
    TOKEN_SEPERRORATOR = ':'
} TokenCode;
```

Write a top-down parser in C.