Question 1  [25 marks]

This question is based on Unit 13.

(i) Construct, in the manner of the tables in Unit 13 Section 2:

(a) the symbol table;  
(b) the array table;  
(c) a real constant table.

for the following definitions and declarations.

```
program tma04q1
const
    duplicate = 2;
    column = 15;
    rate = 9.5;

type
textbox = array [1 .. column] of array ['a'..'f'] of char;
citizen = record
    character: textbox;
    dob: integer
end;

var
car : record
    owner: citizen;
    color: array [1 .. 8] of char
end;
netprice: real;
duplicate, blank: char;
status: Boolean;
```

(ii) By considering the syntax rule for <term>, as given in the Appendix to Unit 13, write down its syntax checking procedure in the manner of Exercise 6.3.

[8]
Question 2  [25 Marks]

This question is based on Section 10 of Unit 14 and assesses your knowledge of semantics.

A type named *string* is to be added to our subset of Pascal. A *string* consists of zero, one, or more characters. For the purposes of this question one new operation is to be supported by the language: *concatenate* represented by the symbol +. This operation *concatenate* is used as an infix operator between either two values of the type *string* or a *string* and a *char*. Thus, if *a* and *b* are *strings*, or one is a *string* and the other is a *char*, the result of

\[ a + b \]

is the string consisting of *b* appended to the end of *a*. It is not permissible to concatenate two *chars*.

The symbol + in the syntax of `<simple expression>`, which previously stood for binary addition, now stands for both binary addition and concatenation.

(i) Draw up the type table for the binary operator + (i.e. including its use for both addition and concatenation). If you use the work ‘anything’ in your table, state what you mean by it. [10]

(ii) What changes, if any, are necessary to the procedure *simpleexpression* given at the top of page 42 in Unit 14 to incorporate the new types and the operator *concatenate*? [5]

(iii) In the manner of part (ii) of Exercise 10.1 write down that part of the function *typecheck* which deals with the operator + (used for both addition and concatenation). [6]

(iv) In Section 10.2 of Unit 14, the syntax diagram for `<expression>` shows that an `<expression>` occurs in one of two forms: either

\[
\begin{align*}
\text{<simple expression>} & \quad \{ \text{case 1} \} \\
\text{or} & \quad \text{<simple expression>} \text{ <relational operator> <simple expression>} \quad \{ \text{case 2} \}
\end{align*}
\]

If the type of the first *<simple expression>* that occurs is denoted *xtype*, and the type of the second *<simple expression>* , if it occurs, is denoted *ytype*, what must be the type of the `<expression>` if no errors are detected. [4]

End of Question 2
Question 3 [25 Marks]

This question is based on Unit 15.

(i) The following definitions and declarations appear in a main program:

```plaintext
const size = 5;
type row = array [0 .. size] of integer;
var total: integer;
    rib: row;
    pattern: record
        piece: row;
        weight: real;
        color: array [1 .. 8] of char
    end;
```

Draw a diagram of the allocation of space in the stack frame for the main program. The diagram should be similar to that given in Figure 11.1 of Unit 15. Assume that integer and character values occupy one storage unit each and that real values occupy two storage units each.

[8]

(ii) Show, by means of a diagram similar to that in the solutions to Exercise 11.4 in Unit 15, the values that will be stored in the address field of the symbol for the variables declared in part (i) above.

[7]

(iii) For the following program show the state of the run-time stack, the static and dynamic links, the run-time display, and current level, each time a procedure is invoked. Stop once the message in procedure p2 has been output. Your solution should be in the form of diagrams similar to those shown in the solution to Exercise 11.8 in Unit 15.

```plaintext
program main;
procedure p1;
procedure p2;
begin
    writeln (‘End of program’)
end;
begin
    p2;
end;
procedure p3;
begin
    p1
end;
begin
    p3
end.
```

[10]

End of Question 3
**Question 4  [25 Marks]**

This question is based on Section 12 of Unit 15.

The following intermediate codes can be used in conjunction to access array elements.

<table>
<thead>
<tr>
<th>code</th>
<th>op1</th>
<th>op2</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td></td>
<td></td>
<td>Get a value from within the stack at a position (i.e. index) given by the number currently at the top of the stack. The value obtained replaces the index as the top item of the stack.</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>y</td>
<td>Used to access elements of arrays. Op2 contains the value of an index to the array table (y). If the element to be accessed is (a[i]), this code assumes that the base address of the array (a) is already held on the stack (at the next top position) and the value of the index (i) is on top of the stack. Code 20 replaces the top two values on the stack by the position (i.e. the index within the stack) of the element (a[i]). The code then uses the base address of (a), the index value (i), and the information in the array table (referenced by (y)) to calculate the stack address of (a[i]).</td>
</tr>
</tbody>
</table>

Here is the relevant part of the symbol table:

<table>
<thead>
<tr>
<th>identifier</th>
<th>type</th>
<th>level</th>
<th>Reference</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>i</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>31</td>
<td>max</td>
<td>1</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>32</td>
<td>n</td>
<td>1</td>
<td>0</td>
<td>28</td>
</tr>
</tbody>
</table>

Here is the array table:

<table>
<thead>
<tr>
<th>index type</th>
<th>element type</th>
<th>element reference</th>
<th>lower bound</th>
<th>upper bound</th>
<th>element size</th>
<th>array size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>integer</td>
<td>integer</td>
<td>0</td>
<td>20</td>
<td>1</td>
<td>21</td>
</tr>
</tbody>
</table>

(i) Using the information in the tables given above, write down the intermediate code which will place the value of \(a[0]\) on top of the run-time stack. Include, in your answer, the code that puts the base address of \(a\) and the index 0 onto the stack.  

[8 marks]

(ii) In a manner similar to (i), write down the intermediate code for the evaluation of the expression:

\(a[i] > max\)

[5 marks]
(iii) Hence, write down the intermediate code for the following piece of program:

\[
\begin{align*}
\text{n} & := 10; \\
\text{max} & := a[0]; \\
\text{i} & := 1; \\
\text{while } \text{i} & \leq \text{n} \text{ do} \\
\text{begin} & \\
\text{if } a[\text{i}] & > \text{max} \\
\text{then} & \\
\text{max} & := a[\text{i}]; \\
\text{i} & := \text{i} + 1 \\
\text{end;} & \\
\end{align*}
\]

[12 marks]