1. [20 marks] In the Algol 68 language whitespace (blanks, tabs, newlines) are used to separate ordinary elements of a program (e.g. reserved words, constants identifiers, etc). Whitespace is also allowed in identifiers (like blanks in file names in Windows and Mac OS).

An identifier in Algol-68 is a sequence of one or more characters that starts with a lowercase letter and continues with lowercase letters, digits or underscores. It may be broken up by spaces, newlines or tab characters.

Examples:
- `real X, initial limit = 7.1 ;`
- `long long real value of pi = 3.14159265358979323846264338327 ;`
- `X := value of pi / initial limit ;`

An Algol 68 compiler needs to capture the names of identifiers for later processing.

a) would it be better to handle whitespace in identifiers during lexical analysis or syntax analysis? Justify your answer.

b) describe a method for processing identifiers containing whitespace.

2. [20 marks] Show the AST data structure that would be generated for the course project language fragment shown below:

```plaintext
forward func F( x : integer ) : boolean
var X, Y[ 10 , -5 .. 5 ] : integer
func F( z : integer ) : boolean
{
    if X < Y[ 2 * z , z ] then
        result true
    else
        Y[ z - 5 , z + 1 ] := X * z
    fi
    result false
}
```
3. [20 marks] The Committee for the Minimization of Programming Languages has made a number of proposals for simplifying the course project language:

a) replace `result` with `return` i.e. in a function use `return expression`

b) remove the reserved word `func` in function declarations

c) add `scope := { declarations }` to the grammar

d) remove the ( ) around the conditional expression `( e ? et : ef )`

Discuss the feasibility of implementing each of these changes.

4. [25 marks] Use the depth first structure alignment algorithm described in the lecture slides to layout the activation record for the function shown below.

```c
1 int solveIt( short start , int end , double X ) {
2     short working[ 10 ] , i, j, k ;
3     struct {
4         char name[7] ;
5         double value ;
6         short link ;
7     } V1, V2, V3[3] ; /* three structure variables */
8     double maxVblue ;
9     struct {
10         char flag ;
11         double lower, upper ;
12         float middle ;
13     } limits[4] ; /* array of structure */
14
15     /* body of solveIt */
16 }
```

You may assume

- the activation record starts with a 4 word block that includes space for the function return value
- scalar parameters are passed by value
- for any type T, `T X[ N ]` declares an array X containing N elements of T
- The size and alignment factors for the basic data types:

<table>
<thead>
<tr>
<th>type</th>
<th>align</th>
<th>size (bits)</th>
<th>type</th>
<th>align</th>
<th>size (bits)</th>
<th>type</th>
<th>align</th>
<th>size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>short</td>
<td>16</td>
<td>16</td>
<td>float</td>
<td>32</td>
<td>32</td>
<td>double</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>int</td>
<td>32</td>
<td>32</td>
<td>char</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. [15 marks] Show the symbol and type table entries that a typical compiler might make for the declarations in Question 5.