1998 Paper 13 Question 6

Compiler Construction

You have been given a new programming language with a C-like syntax, with integer variables and functions and with static binding of free variables. Your manager can parameterise certain aspects of the language, including the following three options:

- For "`int x = e;`" whether the variable `x` has the same l-value of `e` or whether a new l-value is created and initialised to the r-value of `e`. If `e` is only an r-value then a new l-value is created in both circumstances.

- For "`int f(int x) { ... }`" whether the variable `x` is passed by l-value ("by reference") or by r-value ("by value"). If the switch is set to "l-value" and the value passed is only an r-value then a new l-value is created, initialised and passed.

- For "`int f(int x) { ... y ... }`" (where the variable `y` is free to `f`) whether the value of `y` is calculated at the times of its uses (association by l-value) or at the time of the definition of `f` (association by r-value).

As a test of your programming skills your manager asks you to write a program which tells how the language has been parameterised. Do so by printing a 3-digit decimal number where the "hundreds" digit is one or two according to whether the first option is by l-value or r-value respectively, similarly with the "tens" digit for the second and with the "units" digit for the third option.

[10 marks]

Explain the structure of an object module which an assembler or compiler might produce to be processed by a linker. Your answer should include discussion of the various object module features needed to represent the compiled form of the C program:

```
int a[10] = { 2,3,5,7,11,13,17,19,23,29 };
extern int b[10];
extern int g(int);
int f(int y)
{   return g(y) + b[5] + a[6];
}
```

[10 marks]