2004 Paper 8 Question 7

Optimising Compilers

(*a*) Explain the concept of "strength reduction" when applied to loops, illustrating your explanation with the C loop

```
extern int a[M][N];
for (i=0; i<M; i++) for (j=0; j<N; j++) t += a[i][j];
```

Consider the issue of whether strength reduction is sufficient to reduce this to a single loop. [8 marks]

(*b*) The "loop invariant lifting" optimisation says that if an expression is *available* at a point within a loop, and none of its free variables may be updated within the loop, then the expression may be instead computed just before entry to the first iteration of the loop.

Explain the concepts "available" and "computed just before entry" in more detail, focusing your argument by explaining how the following C program could be optimised, expressing the resulting optimised code in a similar syntax to the original.

```
extern int u[100],v[100],w[100];
void f(int n)
{   int i, y = ..., z = ...;
    for (i=5; i<n; i++)
    {    u[i] += 1000/y;
         v[i] += 1000/z;
         p(&y);
         w[i] += 1000/z;
    }
}
```

You need not consider "strength reduction" optimisations here and, because you are expressing the resultant code in C, there is no need to consider register allocation issues. [12 marks]