Maths 1262 quiz 02 answers

Thursday 7/2/13 **Your answers should show all work.**

(1)(i) An array of integers (4 bytes) starts at address 100 and ends at address 299. How many items does it contain? An array is declared

```
double a[41][39];
```

and its starting address is 7654. (ii) What is its size in bytes? (iii)What is the address of `a[21][21]`?

Answers (i) 200/4 = 50 (ii) 41*39*8 = 12792 (iii) 7654 * 39*21*8 + 21*8 = 14374.

(2) Write a routine

```
void reverse ( int n, char x[] )
```

which swaps `a[0]` with `a[n-1]`, `a[1]` with `a[n-2]`, etcetera. Mind you don't swap twice.

```
Answer.
void reverse ( int n, char x[] )
{
  int i,j;
  char c;

  i = 0; j = n-1;
  while ( i < j )
  {
    c = x[i]; x[i] = x[j]; x[j] = c;
    ++i; --j;
  }
}
```

(3) One of the following routine headings is valid, and the other is not, because it doesn't supply enough information to calculate addresses of array entries. Which of them is valid? (Give reasons.)

```
void x ( int n, double a[][10] )
void y ( int n, double a[10][] )
```

Answer. The address formula is $e + i * n * s + j * s$, so n, the number of 'columns' is needed but not the number of 'rows.' Thus `x()` is valid and `y()` is invalid.

(4) Carefully simulate `gcd( 84, 32)`, producing a 'staggered' table of values for the different variables.

```
int gcd ( int m, int n )
{
  int x,y,z;
  x = m; y=n;
  while ( y>0 )
  { z = x % y; x = y; y = z; }
  return x;
}
```

```
 m  n     x       y       z
84 32
         84
                 32
                         20
         32
                 20
                         12
         20
                 12
                          8
         12
                  8
                          4
          8
                  4
                          0
          4
                  0
return 4
```

(5) (Long question, short answer.) In this week's (alternative) programming assignment, you were asked to write a routine which prints both the gcd, call it g, of two (positive) integers m, n and also an expression of g as an integer linear combination $rm + sn$.

It would have been preferable, of course, to have a subroutine `void do_gcd ( ...)` calculate and return g, r, s together, rather than just printing them. This was not asked, because at the time, only call-by-value arguments and local variables had been covered.

Now we have covered more possibilities for declaring variables and subroutine arguments to make them more 'visible.'

Bearing this in mind, give two ways in which a routine `void do_gcd ( ...)` can make g, r, s available to the 'calling program,' i.e., so that after `do_gcd(...)` is called, the values of g, r, s become 'known.'

Answer. (i) Let g, r, s be global variables.

(ii) `do_gcd ( int m, int n, int & g, int & r, int & s)` That is, return g, r, s using reference arguments.