Detector work

A prime number detector requiring only semi-colons? Pretty tight code, you may think. It lends Mike Mudge inspiration to set another poser for investigative readers.

Once upon a time, when Jonathan Cochrane started "messing about" with prime numbers, the first thing he did was to write a function to test whether a number is prime or not. It was easy enough, basically a function of the form:

```
int prime (int x)
{
algorithm;
return PRIME or NOT_PRIME
}
```

While getting the routine working and thinking of what to do next, he decided to try to optimise the prime number routine as much as possible (*haven't we all made this decision? — MM*) and he came up with a prime number detector that requires only semi-colons: pretty tight code, he thought! Can any readers implement a prime number detector satisfying the following specifications?

1. Use any amount of C code you want, but only two semi-colons are to be present.
2. Only allowed to pass one variable to the function, that is, the number to be tested.
3. No pointers are allowed.
4. The function must return a 1 or a 0 depending on prime or composite.
5. Semi-colons within the brackets of a for loop do not count, i.e. for

```
(x=2.3;x<99;x++)
```

^_ _ _ _ _ _ _ ^_ _ _ _ don't count.

6. The following style is also excluded, define semi_colon; Jonathan claims to have tried this on a number of colleagues without finding any solutions (other than his own!).

A different style of investigation, the responses from Numbers Count readers, will be examined with interest. Perhaps other code-based optimisation criteria might be applied?

An exercise in change of number base

Mr P Cowen of Middlesbrough has extended the recent result of JJ Clessa, *viz*. to find a number using the digits 1 to 9 once each only, such that the leading N digits of the number be divisible by N - to different number bases.

His first observation, that the number base must be even (why?) was followed by the use of a Pentium Pro 200 with 64Mb ROM "*which constipated with hard disk over-use at base 34,*" he tells us, but found results for bases 2, 4, 6, 8, 10 and 14. Can any readers extend this investigation, and, if

p276 ➢

possible, find an underlying theory which can be used to dramatically reduce the amount of computation needed to discover such numbers?

Any investigations of the above problems may be sent to me at the address below (see "PCW Contacts"), to arrive by 1st October 1997. All material received will be judged using suitable subjective criteria and a prize will be awarded by *PCW* to the best entry arriving by the closing date. (SAE for return of entry if required, please.) Each contribution should contain brief descriptions of the hardware and software used, together with coding, run times and a summary of the results obtained. General comments on the topics, with references to published or unpublished work in these general areas, would be appreciated.

JAMS: a result

Further to my column which dealt with the subject of JAMS (*Numbers Count, April*) the result X(34732165539) = 876 has been reported by both Mike Bennet (2hr, 11min, 3 sec on an Acorn Risc PC with a StrongARM processor) and by Nigel Backhouse ($4^1/_2$ days on a Pentium 133). So, do not become despondent at the lack of output from this investigation!

Appeals for reference material

Alexander Slack at 106431.2710@compuserve.com would like an *elementary* introduction to Mandelbrot Sets and wonders if there is any software available in QBASIC? Help for a 14-year-old embryo computer scientist would be appreciated.

Perhaps this is not quite in the spirit of Numbers Count, but the author would be interested to receive references to the problem of Tesselations in two dimensions. These need not involve any aspects of computing, although this is clearly a subject where computer graphics skills can be exploited both before and after the underlying maths has been understood.

Close relations

Going back to Numbers Count, December '96, John Sharp observes that the recurrence relation $T_n = 2T_{n-1} - T_{n-4}$ associated with $t^4 = 2t^3 + 1$ (number E) yields the sequence: 0, 1, 1, 1, 2, 3, 5, 9, 16, 29,… for which the ratio of successive terms converges, albeit very slowly, to the Tribonacci Number. Duncan Moore, Nigel Hodges and others found simple algebraic functions for A through I and partial results for the snubdodecahedron, e.g.

```
I = (t(t+2))**1/2, G = (2t+3)**1/2
```

NH also proved, in relation to Problem SL, that: T(2)=128, T(cubes)=12758, T(fourth powers)=5134240,T(fifth powers) =67898771, while T(6th powers) greater than 500 million and T (triangular numbers) = 33.

The worthy prizewinner, however, is Paul Richter of Tunbridge Wells, for a non-sophisticated approach to this investigation. Details from John Sharp at 20 The Glebe, Watford WD2 6LR (or from me).

Going back to your roots

This item in the November '96 column proved to be very popular. The Problem Function lead to a great deal of analysis. Ultimately, Nigel Hodges printed out the two roots to 700 places of decimals showing them to differ in the 647th place. Other analyses included using a program called "Mercury" on a 486DX by Martin Sewell. Duncan Gray refers to p3 of the Excel workbook, *Solutions*. James Lea cites *Numerical Recipes in C* (2nd edition), so this section is very well known.

RF Tindall has been aware of a very fast converging method of approximating to square roots, which is exactly equivalent to the algorithm given, for some time. But he observes that if N is at all large, there are difficulties finding the initial solution.

The worthy prizewinner is Matthew Davies of Luton, who offers an error estimate for the iteration scheme, a generalisation to rational rather than integers, a list of (m_0,n_0) seeds generated using a Turbo Pascal version 6.0 program in the range (1,100). And there's a concluding observation that "*If this technique were to be used as the basis of root calculations on something like an embedded system, I'd be inclined to compile a look-up table of* N....(m_0,n_0) *pairs rather than determine them on-the-fly.*"

■ *Correction: Dec'96 issue, p294, col.3 — for "Scientific American" read "American Scientist".*

PCW Contact

Mike Mudge welcomes correspondence from readers on any subject within the areas of number theory and computational maths, together with suggested subject areas or specific problems for future articles. Email numbers@pcw.co.uk or write to Mike at 22 Gors Fach, Pwyll-Trap, St Clears, Carmarthenshire SA33 4AQ.