Foundations of Programming (Java) 2000

p11q2 FHK

The program is intended to implement the sieve of Eratosthenes and almost does so! Roughly speaking, all values in a boolean array of 600 elements are first set to true. Then, next is set to 2 and for all multiples of next (except 2 itself) the element indexed by the multiple is set to false. The next true element after that indexed by 2 is indexed by 3 and multiples of this value are then treated likewise and so on.

There are three efficiency considerations:

1. Multiples of next are treated in this way only for values of next that are prime.

2. For each such value of next, the first element to be set to false is that indexed by the square of next.

3. The process stops when the square of next exceeds 600. [5 marks]

The bug in the program lies in the condition i<SIZE/next. The problem is that when next=19 the boolean 31<600/19 is equivalent to 31<31 which is false. Accordingly, primes[19x31] is left true. The bug can be fixed by replacing the condition by i*next<SIZE. [4 marks]

If the bug were left unfixed and SIZE and SQRTSIZE were reduced to 150 and 13 respectively, the non-prime 11*13 = 143 would be determined. [4 marks]

The completed program with the bug fixed follows... [7 marks]

```
public class Primes
 { private static final int SIZE=600, SQRTSIZE=25;

   private static void main(String[] args)
    { boolean[] primes = new boolean[SIZE];
      for (int i=2; i<SIZE; i++)
         primes[i] = true;

      int next = 2;
      while (next < SQRTSIZE)
       { for (int i = next; i*next<SIZE; i++)
            primes[i*next] = false;
         do
          { next++;
          } while (!primes[next]);
       }

      int line = 0;
      for (int i=2; i<SIZE; i++)
         if (primes[i])
          { System.out.print(fmtInt(i,5));
            line++;
            if (line%10 == 0)
               System.out.println();
          }
      System.out.println();
    }

   private static String fmtInt(int n, int d)
    { String s = String.valueOf(n);
      while (s.length() < d)
         s = " " + s;
      return s;
    }
 }
```