Training programming exercise #1: Details

Q1: Factoring a cubic polynomial

Given a cubic polynomial $P(x) = x^3 + bx^2 + cx + d$ with integer coefficients, it is possible to find whether it has an integer root and, if it does, to find all real roots. We offer you to write a program that automates this task.

a) Determine whether $P(x)$ has an integer root by trying all divisors of d (including those negative). There are two logically possible cases: either a divisor k of d for which $P(k) = 0$ does exist, or such a divisor does not exist. In the latter case, your program should report that $P(x)$ has no real roots and stop.

b) If $P(k) = 0$ for some divisor k of d, then $P(x)$ can be split into two factors: $(x - k)$ and a quadratic polynomial yet to be determined:

$$x^3 + bx^2 + cx + d = (x - k)(x^2 + b'x + c').$$

Expanding the product, we obtain

$$x^3 + bx^2 + cx + d = x^3 + (b' - k)x^2 + (c' - kb')x - c'k.$$

Comparing coefficients at x^2 and x, we find

$$b' = k + b, \quad \text{then} \quad c' = kb' + c.$$

Thus the quadratic $x^2 + b'x + c'$ becomes fully determined.

c) Having this done, find and print all real roots of $P(x)$ (one root is the k you'd find; others may come from the factored out quadratic polynomial if it has real roots).

[In this question, an array is not needed; this is an exercise on loops, if/else, and the mod (%) operator.]

Consider **two numerical examples** to better understand how the algorithm works.

Example 1. $P(x) = x^3 - 3$. Here $b = c = 0$, $d = -3$. Try all numbers k from -3 to 3 excluding 0:

$k = -3$. Check whether k divides d. — Yes. Check whether $P(-3) = 0$. — No. Proceed.

$k = -2$. Check whether k divides d. — No. Proceed.

$k = -1$. Check whether k divides d. — Yes. Check whether $P(-1) = 0$. — No. Proceed.

$k = 0$. — Skip this value. Proceed.

$k = 1$. Check whether k divides d. — Yes. Check whether $P(1) = 0$. — No. Proceed.

$k = 2$. Check whether k divides d. — No. Proceed.

$k = 3$. Check whether k divides d. — Yes. Check whether $P(3) = 0$. — No. Exit loop. Print "No integer roots".

Example 2. $P(x) = (x+1)(x^2+2x-4) = x^3+3x^2-2x-4$. Here $b=3$, $c=-2$, $d=-4$. The polynomial has been composed as the product of known factors. But let's pretend that factorization is not known. The algorithm below will find it.

Part(a): Try all numbers k from -4 to 4 excluding 0:

$k=-4$. Check whether k divides d. — Yes. Check whether $P(-4)=0$. — No. Proceed.

$k=-3$. Check whether k divides d. — No. Proceed.

$k=-2$. Check whether k divides d. — Yes. Check whether $P(-2)=0$. — No. Proceed.

$k=-1$. Check whether k divides d. — Yes. Check whether $P(-1)=0$. — Yes! Root found. Exit loop

Part (b) begins. Find

$$b' = -1+3 = 2, \quad \text{then} \quad c' = (-1)(2)+(-2) = -4.$$

At this point the algorithm has successfully reconstructed the quadratic factor x^2+2x-4.

Part (c) begins. Solve the quadratic equation

$$x^2+2x-4=0$$

The roots are $-1 \pm \sqrt{5}$. In total, we found 3 roots and can print them:

$$-1, \qquad -1-\sqrt{5} \approx -3.23, \qquad -1+\sqrt{5} \approx 1.23.$$

Note on loop condition in part (a).

You may have some difficulty implementing a loop in part (a). More precisely, the loop has to exit in two circumstances:

— either the value of k exceeds d

— or a root k is found; there is no point checking further values.

There are various ways to combine these conditions. Here is a cheap way. Make a standard counting loop

```
for(k=-d; k<=d; k++)
```

In the *if/else* clause corresponding to the case k divides d and $P(k)=0$, add the line:

```
k=d+1000;
```

(Here 1000 is chosen arbitrarily. Anything greater than 1 would do as well). Next time the loop condition $k \le d$ is checked, it will fail and the loop will exit. Morever, after the loop it will be easy to determine whether it exited due to end of counting range or due to the found root. Put

```
if (k>d+1)
```

after the loop. The value TRUE corresponds to "root found" and your program should proceed to steps (b) and (c).

The value FALSE corresponds to exit upon trying all values of k up to d and failing to find a root. The program should then print "No integer roots" and stop.