> Math 418 : MAPLE Solution of Assignment #4 -- *Your NAME*

Problem 5(a): Using MAPLE's numtheory package:

```
> restart; with(numtheory) :
```

Note: The "restart" command is not necessary, but it is useful for restarting your program if you made multiple mistakes in your computations.

Using the function order in the number theory package:

```
> order(123, 56323);
```

$$4650 \quad (1)$$

Thus, the order of [123] in (Z/56323Z)^* is 4650.

Remark: We can partially check this by observing that

```
> Power(123, 4650) mod 56323
```

$$1 \quad (2)$$

This shows at least that ord(123) | 4650.

>

Problem 5(b): The following program computes the order of a mod m by the naive algorithm:

```
> myord := proc(a, m) local an, n;
    an := modp(a, m);
    if igcd(a, m) ≠ 1 then return(FAIL); fi; #optional
    for n while an ≠ 1 do;
      an := modp(an·a, m); od;
    return(n); end;
```

```
myord := proc(a, m)
    local an, n;
    an := modp(a, m);
    if igcd(a, m) <> 1 then return FAIL end if;
    for n while an <> 1 do an := modp(an * a, m) end do;
    return n
end proc
```

(3)

Testing this for the values of part (a):

```
> myord(123, 56323);
```

$$4650 \quad (4)$$

By inspection we see that this is the same answer as obtained above.
A better way to check this is by using MAPLE's evalb command:

```
> a := 123 : m := 56323 : evalb(myord(a, m) = order(a, m));
```

$$true \quad (5)$$

Next, for a = 1234567 and m = 76543211 we obtain:

```
> a := 12345689 : m := 76543211 : #optional statement
> d := order(a, m);
```

$$d := 12716392 \quad (6)$$

```
> myord(a, m);
```

$$12716392 \quad (7)$$

Thus, both programs give the same answer.

Note: We observe that the program myord(..) took a while to compute this, whereas MAPLE's

order(..) is much faster.
We can **time** how long a given computation takes by using the following construction (or sequence of commands):

```
> settime := time( ) : myord(a, m); time( ) - settime;
```

12716392

9.750 **(8)**

Thus, this computation via myord(..) took 9.7 secs. MAPLE's time for order(..) is:

```
> settime := time( ) : order(a, m); time( ) - settime;
```

12716392

0. **(9)**

Thus, this computation was practically instantaneous.
Presumably MAPLE uses a polynomial time algorithm to compute the order, which exists if phi(m) is known. This is feasible here since phi(m) can be easily computed and factored:

```
> phi(m);
```

76298352 **(10)**

```
> ifactor(%);
```

$(2)^4\ (3)\ (13)\ (122273)$ **(11)**

Remark: In the above algorithm we used the obvious recusion relation: a^k = a^(k-1)*a.
If, instead, you had calculated a^k fresh at each step, then we get a *much slower* algorithm:

```
> myord_slow := proc(a, m) local n;
    if igcd(a, m) ≠ 1 then return(FAIL); fi; #optional
    for n while power(a, n) mod m ≠ 1 do; od;
    return(n); end;
```

```
myord_slow := proc(a, m)
    local n;
    if igcd(a, m) <> 1 then return FAIL end if;
    for n while power(a, n) mod m <> 1 do end do;
    return n
end proc
```
(12)

```
> settime := time( ) : myord_slow(a, m); time( ) - settime;
```

12716392

20.951 **(13)**

Thus, this took almost 21.0 secs (more than twice as long).
Note that the above algorithm invoked the power-mod algorithm. If we didn't use this, then not only would the algorithm take too long but we would also get overflow error.

```
> myord_superslow := proc(a, m) local n;
    if igcd(a, m) ≠ 1 then return(FAIL); fi; #optional
    for n while a^n mod m ≠ 1 do; od;
    return(n); end;
```

```
myord_superslow := proc(a, m)
    local n;
    if igcd(a, m) <> 1 then return FAIL end if;
    for n while a^n mod m <> 1 do end do;
    return n
end proc
```
(14)

> $settime := time(\,) : myord\_superslow(a, 48611); time(\,) - settime;$

24305

70.809 **(15)**

Thus, even with this extremely small m = 48711, finding ord_m(a) took 70.8 secs.
To compute ord_m(a) with the given m would take many **WEEKS** (more than a **YEAR!**)
(provided thatf we don't encounter an overflow error).
To see this, note first that compute a^n mod m for n > 300,000 takes at least 1/10 sec:

> $settime := time(\,) : a^{300000} \bmod m; time(\,) - settime;$

18878889

0.125 **(16)**

Thus, to compute a^n mod m for n from 300001 to d =19074588 would take at least

> $t := (d - 300001) \cdot \left(\frac{1}{10} \right);$

$$t := \frac{18774587}{10} \quad \textbf{(17)}$$

seconds. This equals 31291min or 513 hours or 21.4 **days,** as the following computation shows:

> $evalf\left(\frac{t}{60} \right), evalf\left(\frac{t}{3660} \right), evalf\left(\frac{t}{24 \cdot 3660} \right);$

31290.97833, 512.9668579, 21.37361908 **(18)**

In actual fact, it takes even longer: for n > 5,000,000 we need at least 2.5secs per computation:

> $settime := time(\,) : a^{5000000} \bmod m; time(\,) - settime;$

69641862

2.948 **(19)**

Thus, to compute a^n mod m for n from 5,000,000 to d =19074588 would take at least 400 **DAYS:**

> $t2 := (d - 5000000) \cdot 2.5 : evalf\left(\frac{t2}{60} \right), evalf\left(\frac{t2}{3660} \right), evalf\left(\frac{t2}{24 \cdot 3660} \right);$

$5.864411667\ 10^{5}$, 9613.789617, 400.5745674 **(20)**

Finally, even if we had computed that far (i.e. for 400 days > 1 year), then eventually we would have gotten an overflow error:

> $a^{d} \bmod m;$

Error, cannot allocate memory (size=142639104)

>