Matthew Soicher
z5160737

Question 1

Instead of computing M^n as simply M multiplied by itself $n-1$ many times, we can rather approach it using this idea:

We split the computation by using a base-2 binary representation of the exponent, n.

If we have 5^7, for example, we express the component in base-2 binary as 5^{111}.

We know that in all cases, the exponent n will always have, at worst case, $(\log_2 n)$ + 1 many digits.

We can verify this in our case: Our exponent is 111, which is 3 digits.

For cases which do not yield an integer, we take its floor

And so, $(\log_2 7)\ +\ 1\ =\ 3$

We can see via this representation, that 5^{111} can be computed as a multiplication of each individual binary spot, in our case: $5^4 * 5^2 * 5^1$.

Since we don't need to compute the base, 5^1 which is just M, we need to perform only $\log_2 n$ multiplications, again taking its floor for non-integer results.

i.e. = $(\log_2 7) = 2$

To calculate these multiplications, we notice each element that we compute is the square of the previous element.

$$5^1 = 5$$
$$5^2 = (5^1)^2 = 5^2 = 25$$
$$5^4 = (5^2)^2 = 25^2 = 625$$

And so, by doing a maximum of $(\log_2 n)$ multiplications (taking the floor for non-integer results), we can compute our answer by multiplying the relevant powers.

In our example, $5^7 =\ 625 * 25 * 5 = 78{,}125$.

The complexity of the program is $O(\log n)$: The computation of the powers is at most $\log n$ and the multiplication $\log n$ as well.

For the algorithm, we divide the problems into subproblems of size n / 2 and call them recursively. Two cases are required; for if n is even, or if n is odd (we take its floor)

See pseudocode on next page.

Matthew Soicher
z5160737

Pseudocode

```
int question1(int M, int n)

   if (n == 0)              // base case
      return 1

   else if (n % 2 == 0) // n is even
      return question1(M, n/2) * question1(M, n/2)

   else                     // n is odd
      return M * question1(M, n/2) * question1(M, n/2)

end
```

The reason that the odd case returns *M * question1(M, n/2) * question1(M, n/2)*, is because all odd powers of n will have a 1 as their rightmost bit, representing M^1, so the result will always need to be multiplied by M. Even cases of n, will not.