Part D: Additional problems (10 points)

In this section, we'll write some functions that were referred to in the lectures but not defined.

1. `is_prime`

[5 points]

In lecture 5, we referred to a function called `is_prime` which returns `true` if an integer is a prime number and `false` otherwise. Write this function. For the purposes of the function, consider all negative numbers, zero, and 1 to be non-prime (and thus return `false` instead of *e.g.* raising an exception). Your function doesn't have to be maximally efficient, but it should have a time complexity of at most $\Theta(\sqrt{n})$ for input n.

Hints

For any input n you only have to check if numbers up to $\sqrt{n}$ can divide n, since if they can't, no larger number can either, and so n must be prime.

The OCaml builtin function `sqrt` works on `float`s only; you may find the functions `float_of_int` and `int_of_float` to be useful.

2. `smallest_prime_factor`

[5 points]

In lecture 5, we also referred to a function called `smallest_prime_factor` which returns the smallest prime factor of a composite (non-prime) positive integer. Write this function. (You can use the `is_prime` function you defined above in the definition.) If the input number is prime or less than 2, raise an exception using the `invalid_arg` function. Again, we're not worried about maximal efficiency here, but your function must work properly.