You have 50 minutes. You may use one prepared 8.5 × 11 sheet of paper. All questions concern the C++ programming language. **Check your answers.**

1. Some integers are *perfect* and some are not. The definition of a perfect integer is not important for this problem. You have been given a definition of function `isPerfect(n)` that returns true if positive integer `n` is perfect and returns false if `n` is not perfect. You can use that function.

   Write a C++ definition of function `numPerfect(m, n)`, which returns the number of integers `x` where `m ≤ x ≤ n` and `x` is perfect. Assume that `0 < m ≤ n`.

   Use a scan algorithm for this function. **Use a loop. Do not use recursion. Do not use arrays or strings or any data structure from the Standard Template Library.**

   Plan the loop by choosing reasonable example values of `m` and `n` and a reasonable example positing which of `m, m + 1, . . . , n` are perfect.

   A heading is given.

   ```cpp
   int numPerfect(const int m, const int n)
   ```
2. Again assume that function isPerfect(n) is available.

Write a C++ definition of function somePerfect(m, n), which returns true if at least one of the numbers in list \( m, m + 1, m + 2, \ldots, n \) are perfect, and returns false otherwise.

Use a search algorithm for somePerfect. As soon as your algorithm finds that a number is perfect, it must not call isPerfect again.

Use a loop for this problem, not recursion. Do not use arrays or strings or any data structure from the Standard Template Library. A heading is given.

```cpp
bool somePerfect(const int m, const int n)
```

3. Not all functions change the values of variables. Function next(n) for the hailstone sequence is defined to return \( \frac{n}{2} \) if \( n \) is even and \( 3n + 1 \) if \( n \) is odd. Write a definition of next(n). Its body must not change the value of any variable. A heading is given.

```cpp
int next(const int n)
```