Problem 6—Guessing Game

Professor Plum routinely drives a van-load of CS students to MICS. To keep awake, the "lucky" student in the front passenger seat gets to play the hidden-number guessing game with Professor Plum. They agree on the largest possible value *n* and the student selects a hidden number from the set of integers {1, 2, ..., *n*}. The professor repeatedly tries to guess the hidden number with the student responding with one of three possible answers:

- "Your guess is lower than the hidden number", or
- "Yes, that's it!", or
- "Your guess is higher than the hidden number".

On one MICS trip, a student got tired of Professor Plum doing a "binary search" of the range to minimize the number of guesses in the worst case. The student asked Professor Plum to consider a related optimization problem: "Given the value of *n*, what is the *optimal strategy* **that minimizes the sum of all incorrect guesses** leading up to the hidden number in the worst possible case."

At first Professor Plum did not understand, so the student talked through some examples.

If *n*=3, the best we can do is obviously to guess the number "**2**". The answer will immediately lead us to find the hidden number (at a total cost = 2). The total cost is 2 because the one incorrect guess is 2, and not because we need 2 guesses.

If *n*=8, we might decide to use a "binary search" type of strategy: Our first guess would be "**4**" and in the worst case the hidden number is higher than 4, so we will need one or two additional guesses. Let our second guess be "**6**". If the hidden number is still higher than 6, we will need a third guess in order to discriminate between 7 and 8. Thus, our third guess will be "**7**" and the total cost for this worst-case scenario will be 4+6+7=**17**.

Professor Plum begins to understand when the student points out that we can improve considerably the worst-case cost for *n*=8, by guessing "**5**"first. If we are told that the hidden number is higher than 5, our second guess will be "**7**", then we'll know for certain what the hidden number is (for a total cost of 5+7=**12**). If we are told that the hidden number is lower than 5, our second guess will be "**3**" and if the hidden number is lower than 3 our third guess will be "**1**", giving a total cost of 5+3+1=**9**. Since **12**>**9**, the worst-case cost for this strategy is **12**. That's better than what we achieved previously with the "binary search" strategy; it is also better than or equal to any other strategy. So, in fact, we have just described an optimal strategy for *n*=8.

Let C(*n*) be the worst-case cost achieved by an optimal strategy for *n*, as described above.
Thus C(1) = 0, C(2) = 1, C(3) = 2, C(8) = 12, C(20) = 49, and

Professor Plum want you to write a program that takes as input a single positive integer X, and calculates

INPUT SPECIFICATION– File name "prob6.in"

The input file contains a single line with a positive integer X. You program should find

OUTPUT SPECIFICATION.

The output file should contain a single line with a single integer answer corresponding to the input X.

SAMPLE INPUT.

20<EOLN>
<EOF>

SAMPLE·OUTPUT.

402<EOLN>
<EOF>