Direct Method: -
In this directly called exam variable and got the first 10 result as known and then double that to get answer which made my worst marks to be 10 but knew that it can be optimized further so tried a variety of ways to do that.

Occurrences Approach: -
In this, I identify the most frequently occurring pairs and triplets and use them to estimate a code in which can get best possible solution with its frequency of both. Made pairs such as (0,1), (1,0), (0,0), (1,1) and similarly for triplets (0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,1,0), (0,1,1), (1,1,1) etc, but with this approach there is still some cases where can't get the best result possible.

Grouping Approach / Bit-wise Approach: -

- Two's / Pair Grouping: -

Started with new approach of grouping of two consecutive indexes (0,1)(2,3).....(18,19), which maid these 20 answers into 10 and took 0 and 1 from pair to get a 10 digit code to get the answer but still the worst mark were 10 as in some cases as some cases have indexing as (0,1) and (1,1) in same place causing it to have 50% accuracy only.

- Three's Grouping: -

Started make groups of 3 sets of answer bits indexes (0,0,0), (0,0,1) (0,1,0), (0,1,1), (1,0,0) …. (1,1,1) etc. which gives us the worst-case result of 12 which is better than our previous methods which shows odd grouping is better than the even number to prevent domination of either 0 or 1 in grouping.

- Four's Grouping: -

This can be skipped as it is an even number grouping and can cause domination of 0 or 1 causing deviation from answer and making worst possible result as 10 or 50% and no improvement can be done.

- Five's grouping: -

This implementation also produces a worst case of 12. As it consists of set of 5 bits, so it has some cases such as (0,0,0,1,1) and (0,0,0,0,0) which causes accuracy to reduce and causes 12 to be the worst possible.

So that means that three's grouping can also give an answer of 12 where 2 less bits are required to get an answer. This causes rest of the bits to be unused and which can be further processed and used.

Unused Bits: -
Will try to utilize the remaining bits left in each case to get the best possible outcome in all cases. For three's grouping we have 4 spare bits. In the case of grouping threes, with 4 spare bits, mapping the last 4 bits directly to the final four questions ensures a worst case of 14, an improvement from the earlier 12. However, there's room for improvement. For the final grouping of three at indexes 15, 16, and 17, overwriting two with uncompressed answers is wasteful. Reclaiming one bit and keeping it as an uncompressed mapping result in a worst case of 15 (5(2) + 5).
Similarly, we will try it with case of five's grouping but it will give us more than 20 bits which is not possible. So, the best way is to remove 7 uncompressed mapping from 20 bit which will have 13 bits remaining.
Now split the remaining 13 bit in between two sets of fives grouping and one set of three's grouping to maximum utilization and best result possible.
So, the worst case will become 2(3) + 1(2) + 7 = 15.

Other Combinations possible for the best result 15 so far are: -

- 2 five's grouping, 1 three's grouping, and 7 one to one mappings.
- 5 three's grouping and 5 one to one mappings.

So, the combination we used above is the simplest and easy to use.

Conclusion: -
In this the use of hamming code might have given better solution and a higher result which is possible i.e. 17 instead of 15 by using hamming distance as the metric of measurement. Solving it with error correction and decode it then will yield better results than mapping the bits one to one and place it in best possible way. However due to research and solving it in limited time unable to get better result than this and explore it further so submitted this solution which can be used as an alternate to get the result.
Here, we are assuming that there is no external interference when the message is sent.

Reference: -
https://github.com/rlamprell/compression_and_encoding