Find all natural numbers belonging to the segment [35,000,000; 40,000,000], which have exactly five different odd divisors (the number of even divisors can be any). In the answer, list the numbers found in ascending order

```
void oddDivisor(int number, int secondNum) {
  int count;                                  // counts number of divisors
  for (int i = number; i <= secondNum; i++) {  // first loop for checking numbers between the two given
     count = 0;                               // starts counter at 0 inside first loop
     for (int j = 1; j <= i; j++) {           // second loop looks at each number's divisors for odd
        if (i % j == 0 && j % 2 != 0 ) {      // checks if divisor is divisor & odd
           count++;
        }
     }
     if (count == 5) {
        std::cout << i << std::endl;
     }
  }
}
```

324

1, 2, 3, 4, 6, 9, 12, 18, 27, 36, 54, 81, 108, 162, 324

2*2*3*3*3*3

729

81=3^4
162=2*3^4
324=4*3^4
625=5^4
648=8*3^4

number=2^k*prime^4

135/3=45 counter++
45/3=15 counter++
15/3=5 counter++
5 odd counter++

1, 3, 5, 9, 15, 27, 45, 135

3*5

1,3,5,3*5 - 4 odd devisors

3*3*5
1,3,3*3,5,3*5 -6 odd devisors

7*7*7*7*2^p
1 ,7, 7*7 ,7*7*7 ,7*7*7*7 -5 odd devisors

```
void betterOddDivisor(int number, int secondNum) {
  int temp;
  int counter;
  int flag;
  int flagBreak;
  for (int k = number; k <= secondNum; k++) {
     temp = k;
     counter = 0;
     double root = sqrt(temp);
     flag = 0;
     flagBreak = 0;
     for (int i = 2; i <= root; i++) {
        if (temp % i == 0) {
           if (i % 2 != 0) {
              if (flag == 0) {
                 flag = i;
                 counter++;     // starts counting 1st odd divisor instance
              }
              else if (i != flag) {
                 flagBreak = 1;
                 break;         // this stops the  'for' loop, not just the 'ifs' (encountered a different odd divisor)
              }
              else if (i == flag) {
                 counter++;       // encountered another odd divisor similar to 1st instance
              }
           }
           temp = temp / i;
           root = sqrt(temp);
           i--;
        }
     }
     if (flagBreak == 0) {
        if (temp % 2 != 0) {   // adds last encounter of the divisor for the '5th' count
           if (temp == flag) {
              counter++;
           }
           else if (temp != flag) {
              flagBreak = 1;
           }
        }
     }
     if (flagBreak == 0) {
        if (counter == 4) {
           std::cout << k << std::endl;
        }
     }
  }
}
```